NEC

USER'S MANUAL
EVAKIT-7764

NEC ELECTRONICS (EUROPE) GMBH

USER'S MANUAL

EVAKIT-7764

1/85 V1.0

Table of Contents

Page

CHAPTER 1 OUTLINE

1.1 Outline

The EVAKIT-7764 is a development support system to develop the μPD7764 speech recognition processor efficiently.

Since the μPD7764 does not have a ROM for programming, powerful functions for maintenance and management of the program are required for development of software.

The EVAKIT-7764 is provided with functions with which the contents of the instruction RAMs, data RAM, or the internal registers can be easily displayed or modified, or such functions as to stop the program execution at the required address or after the required number of executions to display the contents of the internal registers. Furthermore, the EVAKIT-7764 has an on-line assembly and disassembly functions. With these functions the EVAKIT-7764 offers, debugging the μPD7764 program can be efficiently done.

In addition, by connecting the EVAKIT-7764 to the user system, debugging the program developed by the user system can be also performed.

Another feature of the EVAKIT-7764 is that it is provided with LOAD and SAVE commands that input/output data to/from the instruction RAMs, data RAM, and internal registers. If the host system has a control program that controls the RS-232C interface, therefore, software development with repeatability will become possible. For effective and efficient application of the LOAD and SAVE commands, the EVC-7764, which is used with the MD-080FD-10 as the host system, is optionally available.

1.2 Features of EVAKIT-7764

(1) The contents of the instruction RAMs, data RAM, and internal registers can be easily displayed or modified.

(2) By setting a break point (as an address or the number of times of program execution), the program execution can be stopped at the required address or can be repeated the required number of times.

(3) The contents of the internal registers can be displayed when the program execution passes the set break point.

(4) Since the RS-232C interface is provided as a serial interface, connecting the EVAKIT-7764 to a console is easy.

(5) The user system can be easily connected with the emulation cable. By doing so, development and debugging of existing systems are possible.

(6) The on-line assembly and disassembly functions allow efficient software debugging.

(7) As the pattern memory for the μPD7764, 64K bytes are available.

(8) In combination with the EVC-7764 (with the MD-080FD-10 serving as the host system), programs and data can be transferred to/from the host system, thereby enabling program development with repeatability.

1.3 Functions of EVAKIT-7764

The monitor commands of the EVAKIT-7764 are listed in Tables 1-1 and 1-2.

Table 1-1 shows whether the contents of each register can be referenced or manipulated when each command is executed.

Table 1-2 shows whether the contents of each register will be held after execution or emulation, or during emulation of each command.

The keys to understanding Table 1-1 are as follows:

(1) The column under the heading "DUMP" indicates whether the contents of each register can be displayed when the DUMP command is executed.

(2) The column under the heading "CHANGE" indicates whether the contents of each register can be modified when the CHANGE command is executed.

(3) The column under the heading "SAVE & LOAD" indicates whether data can be input/output to/from each register via the RS-232C interface when the SAVE or LOAD command is executed.

(4) The meanings of the symbols are as follows:

- : Not affected
- o : Can be manipulated
- x : Cannot be manipulated

Table 1-1 List of EVAKIT-7764 Functions

Command / Register /memory	DUMP	CHANGE	SAVE & LOAD
SR	—	—	×
DR	—	—	×
IR	●	●	●
PC1	●	×	●
GR0	●	●	●
GR1	●	●	●
GR2	●	●	●
GR3	●	●	●
GR4	●	●	●
GR5	●	●	●
GR6	●	●	●
GR7	●	×	●
GARG	●	●	●
MC	●	×	●
DRG0	●	●	●
DRG1	●	●	●
IOSR1	●	●	●
IOSR2	●	●	●
PCNTA	●	●	●
PCNTB	●	●	●
PC2F	●	×	●
BAR	●	●	●
LCNT	●	●	●
RCNT	●	●	●
WCNTA	●	●	●
WCNTB	●	●	●
IM1	●	●	●
IM2	●	●	●
RAM-A	●	●	●
RAM-B	●	●	●
RAM-G	●	●	●
PMEM	●	●	●

The keys to understanding Table 1-2 are as follows:

(1) The column under the heading "At GO execution" indicates whether the contents of each register can be manipulated when the GO command is executed.

(2) The column under the heading "At passing break point" indicates whether the contents of each register can be dumped by setting parameter "/D" in the same way as at execution of the DR command when the program execution passes the break point during GO command execution (break point setting).

(3) The column under the heading "At break completion" indicates whether the contents of each register can be retained when the GO command (break point setting) is executed (i.e., when the break condition is satisfied).

(4) The column under the heading "At GO forced completion" indicates whether the contents of each register can be retained when the emulation is aborted by inputting "/" during execution of the GO command.

(5) The column under the heading "At USER completion" indicates whether the contents of each register can be retained when the mode is changed from user to EVAKIT by inputting "/" during execution of the USR command.

(6) The meanings of the symbols are as follows:

- : Not affected
- ● : Retained
- x : Not retained
- *1: Can be referenced by specifying a parameter during execution of the GO command.
- *2: Can be referenced or manipulated according to change in the ROM flag of the SR register during execution of the GO command.

*3: Only the contents of F (flags) of the PC2F register are retained.

*4: The contents are retained only when the contents of either of the instruction memories (IM1 or IM2) satisfy the break condition.

*5: The contents input to the μPD7764 are retained.

*6: The contents input to the μPD7764 from the user system are retained.

*7: Either the contents input from the user system to the μPD7764 or those when the mode is changed to the EVAKIT mode are retained.

Table 1-2 List of EVAKIT-7764 Functions

Condition / Register /memory	At GO execution	At passing break point	At break completion	At GO forced completion	At USER completion
SR	*1	—	—	—	—
DR	*2	—	—	—	—
IR	—	×	●	●	●
PC1	—	●	●	×	×
GR0	—	●	●	●	●
GR1	—	●	●	●	●
GR2	—	●	●	●	●
GR3	—	●	●	●	●
GR4	—	●	●	●	●
GR5	—	●	●	●	●
GR6	—	●	●	●	●
GR7	—	●	●	×	×
GARG	—	●	●	●	●
MC	—	●	●	×	×
DRG0	—	●	●	●	●
DRG1	—	●	●	●	●
IOSR1	—	●	●	●	●
IOSR2	—	●	●	●	●
PCNTA	—	●	●	●	●
PCNTB	—	●	●	●	●
PC2F	—	*3	*3	×	×
BAR	—	●	●	●	●
LCNT	—	●	●	●	●
RCNT	—	●	●	●	●
WCNTA	—	●	●	●	●
WCNTB	—	●	●	●	●
IM1	—	—	*4	*5	*6
IM2	—	—	*4	●	*7
RAM-A	—	—	●	●	●
RAM-B	—	—	●	●	●
RAM-C	—	—	●	●	●
PMEM	—	—	●	●	●

1.4 System Configuration

The EVAKIT-7764 consists of two boards: an emulator board and a CPU board.

The emulator board is provided with functions to emulate the μPD7764, 64K-byte pattern memory, etc., whereas the CPU board has functions to transfer data to/from the console, control the emulator board, etc.

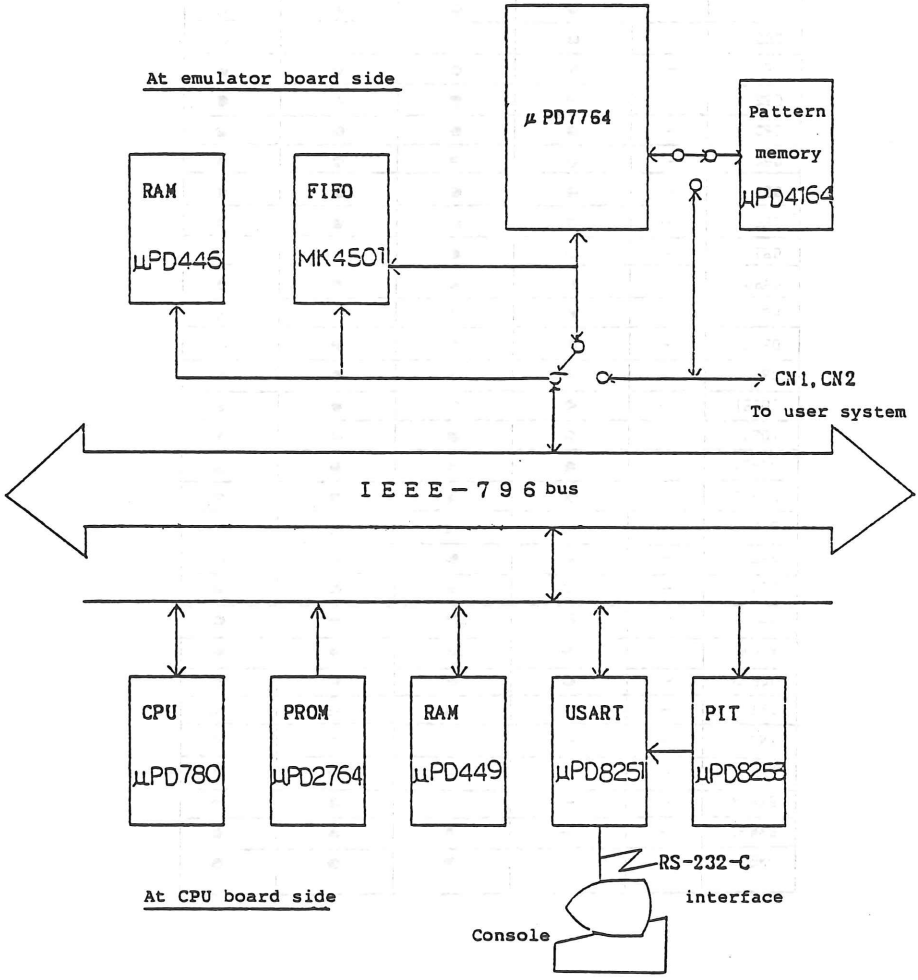


Fig. 1-1 EVAKIT-7764 System Configuration

1.5 Hardware Specifications

1.5.1 Specifications

Model name	:	EVAKIT-7764
Main LSI	:	Emulator board
		CPU : μPD7764 (x1)
		FIFO : MK4501 (x1)
		Pattern memory : μPD4164 (x8)
		RAM : μPD446 (x4)
		CPU board
		CPU : μPD780 (x1)
		PROM : μPD2764 (x4)
		RAM : μPD449 (x2)
		SIO : μPD8251 (x1)
		Timer/counter : μPD8253 (x1)
Operating temperature	:	10 to 40°C
Operating humidity	:	90% RH max. (without condensation)
Storage temperature	:	-45 to 65°C
Supply voltage	:	+5V ±5% (5A max.)
		+12V ±5% (0.1A max.)
		-12V ±5% (0.1A max.)
Dimensions of boards	:	18.5 x 32.5 x 7.7mm
Weight	:	770g
Accessories	:	Emulation cable (50cm)
		RS-232C interface cable (150cm)
		Power cable (50cm, x4)

1.5.2 Appearance and construction of EVAKIT-7764

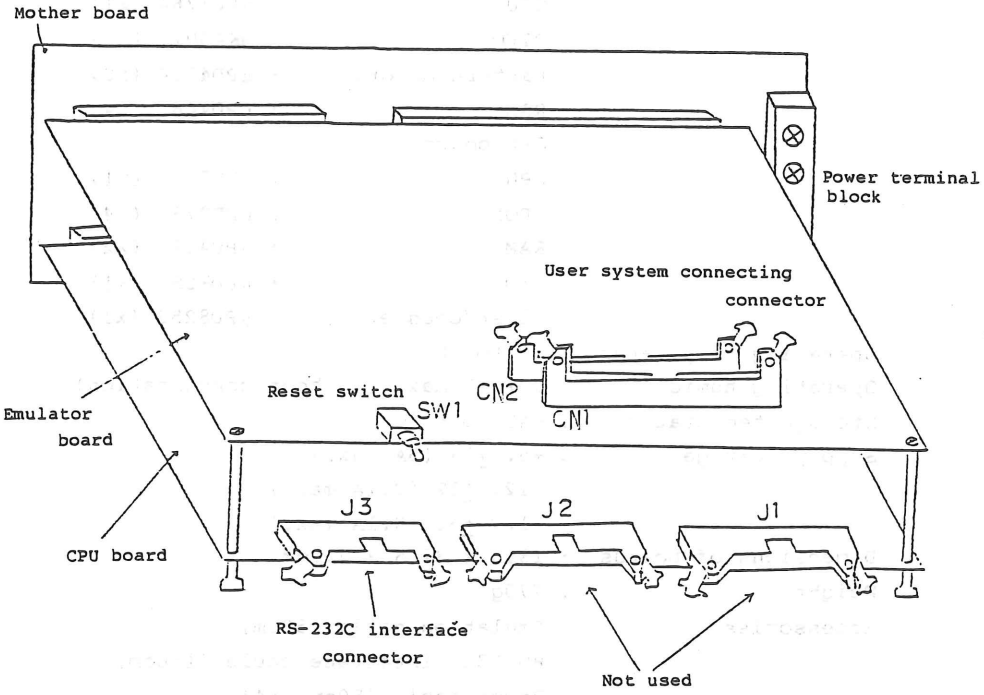


Fig. 1-2 Appearance and Construction of EVAKIT-7764

1.5.3 Component layout on emulator board

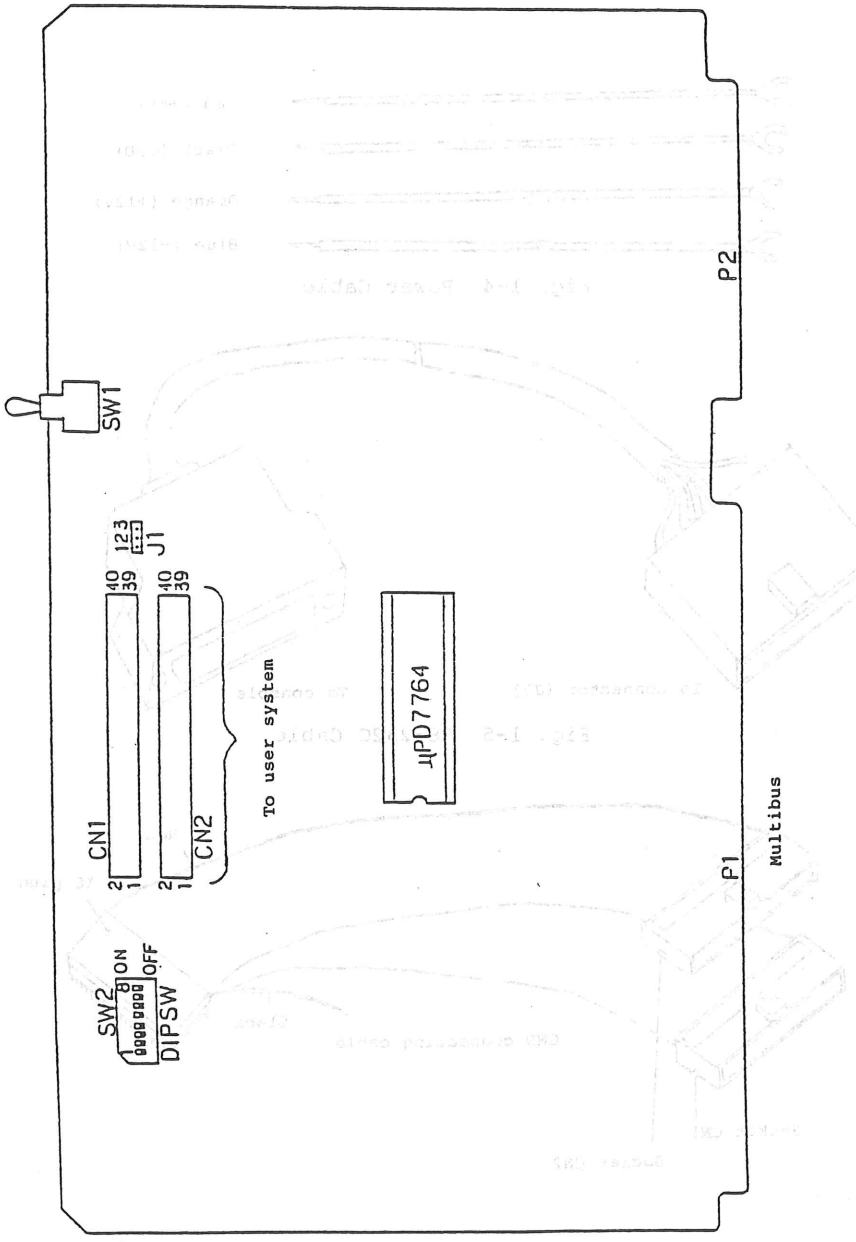


Fig. 1-3 Component Layout on Emulator Board

1.5.4 Accessories

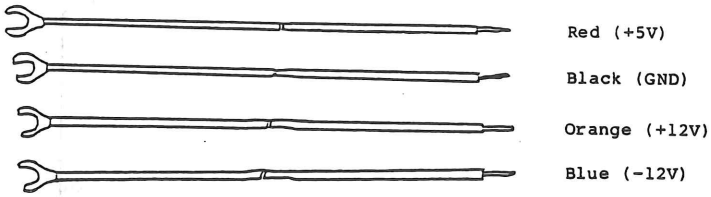


Fig. 1-4 Power Cable

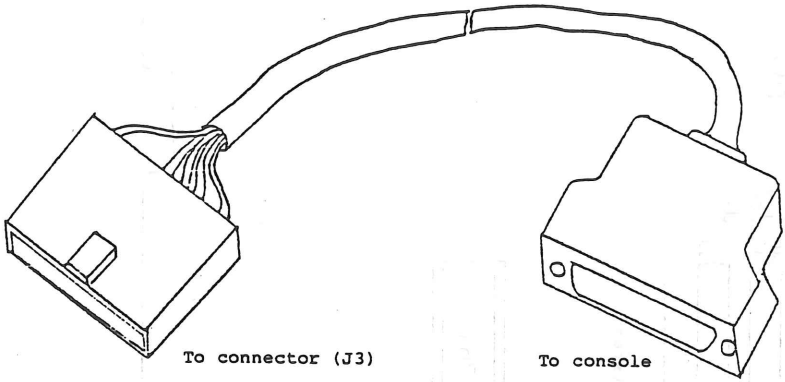


Fig. 1-5 RS-232C Cable

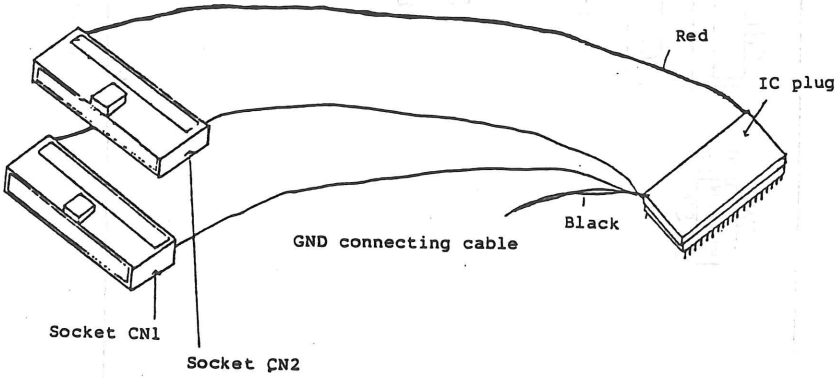


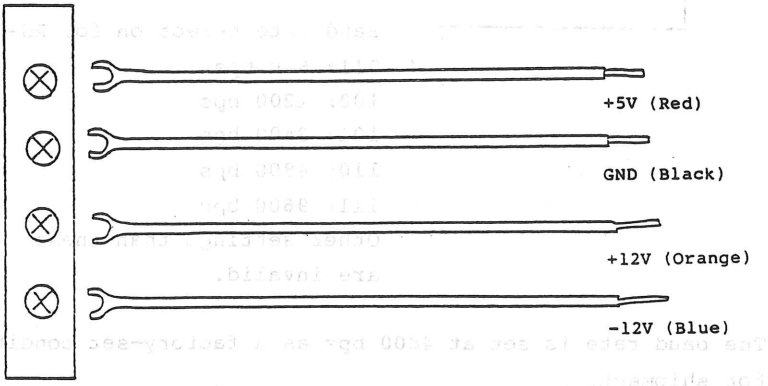
Fig. 1-6 Emulation Cable

CHAPTER 2 SETTING OF EVAKIT-7764

2.1 Before Power Application

2.1.1 Power cable connection (+5, GND, +12, -12V)

Connect the power cable attached as an accessory to the power terminal block as shown in Fig. 2-1 and apply the rated voltage.



*Perform the common grounding on each of the power cables.

Fig. 2-1 Connection of Power Terminal Block

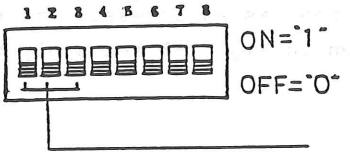
2.1.2 Specifications of RS-232C and its connection to console

For the EVAKIT-7764, use a console with the following specifications.

Full duplex, asynchronous

Start bit	: 1 bit
Data length	: 8 bits
Parity	: None
Stop bit	: 2 bits
Baud rate	: 600/1200/2400/4800/9600 bps

The baud rate is selected according to the setting of DIP switch (SW2) pins 1 to 3.



Baud rate selection for RS-232C
011: 600 bps
100: 1200 bps
101: 2400 bps
110: 4800 bps
111: 9600 bps
Other settings than these are invalid.

*The baud rate is set at 4800 bps as a factory-set condition for shipment.

As a connector, connector J3 (26-pin) on the CPU board is used. Signals listed in Table 2-1 are transferred via this connector.

Table 2-1 Connector (J3) Pin Assignment

Connector (J3) pin No.	Signal name	Direction (EVAKIT ------ console)	Console pin No.
26	FG		1
24	TxD	←	3
22	RxD	→	2
20	RTS	←	4
18	CTS	→	5
16	DSR	→	6
14	SG		7
13	DTR	←	20

The other pins than the above are opened.

Description of signals

TxD (Transmitted Data)	: Data signal transferred from the console to the EVAKIT
RxD (Received Data)	: Data signal transferred from the EVAKIT
RTS (Request To Send)	: Signal to request data of the EVAKIT
CTS (Clear To Send)	: Enables the console to transfer data
DSR (Device Status Ready)	: Used as the output condition for the RTS signal by the console. This signal is normally active.
DTR (Data Terminal Ready)	: Indicates that the console becomes active. The EVAKIT will not operate unless the console is active.
SG (Signal Ground)	: GND for signal
FG (Frame Ground)	: GND for frame

Connect the RS-232C cable attached as an accessory to the connector (J3) on the CPU board as shown in Fig. 2-2.

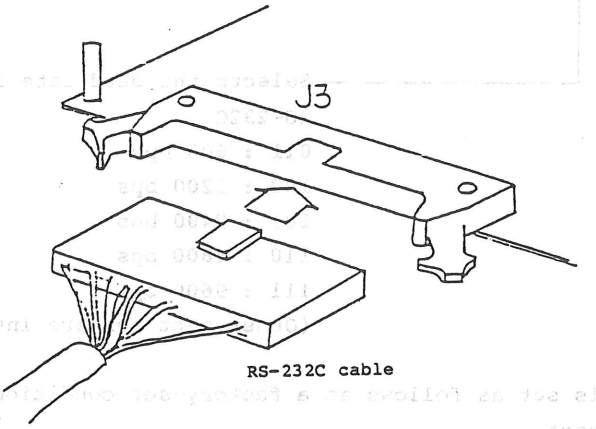
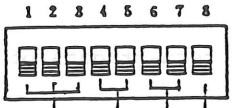


Fig. 2-2 RS-232C Cable Connection

2.1.3 Setting of DIP switch (SW2)

DIP switch SW2 is used to specify the operating states of the EVAKIT-7764. Set this switch as follows:



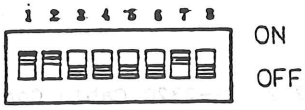
Selects the pattern memory for the μPD7764
0 : The RAM on the EVAKIT is used.
1 : The RAM on the user system is used.

Selects the addresses of the work RAM
01 : 8000H to 9FFFH (The other settings are inhibited.)

Not used

Selects the baud rate of the RS-232C
011 : 600 bps
100 : 1200 bps
101 : 2400 bps
110 : 4800 bps
111 : 9600 bps
(Other settings are invalid.)

*SW2 is set as follows as a factory-set condition for shipment.



2.1.4 Clock Selection (J1)

The clock for the μPD7764 is selected by jumper J1 on the emulator board.

When jumper points 1 and 2 are connected as shown in Fig. 2-3, μPD7764 uses the clock on the EVAKIT.

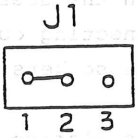


Fig. 2-3 Connection of J1 to Use Clock on EVAKIT

When jumper points 2 and 3 are connected as shown in Fig. 2-3, μPD7764 uses the clock on the user system.

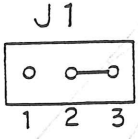


Fig. 2-3 Connection of J1 to Use Clock on User System

Table 2-3 Setting of J1

Setting	Function
1 to 2	The clock on the EVAKIT is used.
2 to 3	The clock on the user system is used.

*Jumper points 1 and 2 are connected, in other words, the clock on the EVAKIT is specified as a factory-set condition for shipment.

2.1.5 Connecting EVAKIT-7764 to user system

Connect the EVAKIT-7764 to the user system as shown in Fig. 2-5 with the emulation cable attached as an accessory.

Connect one end of the cable to the user system connecting connectors (CN1 and CN2) on the emulator board and the other end to the μPD7764 socket on the user system.

Connect the user system connecting connectors (CN1 and CN2) on the emulator board to the sockets of the emulation cable as shown in Fig. 2-6.

Connect the IC plug of the emulation cable to the μPD7764 on the user system as shown in Fig. 2-7.

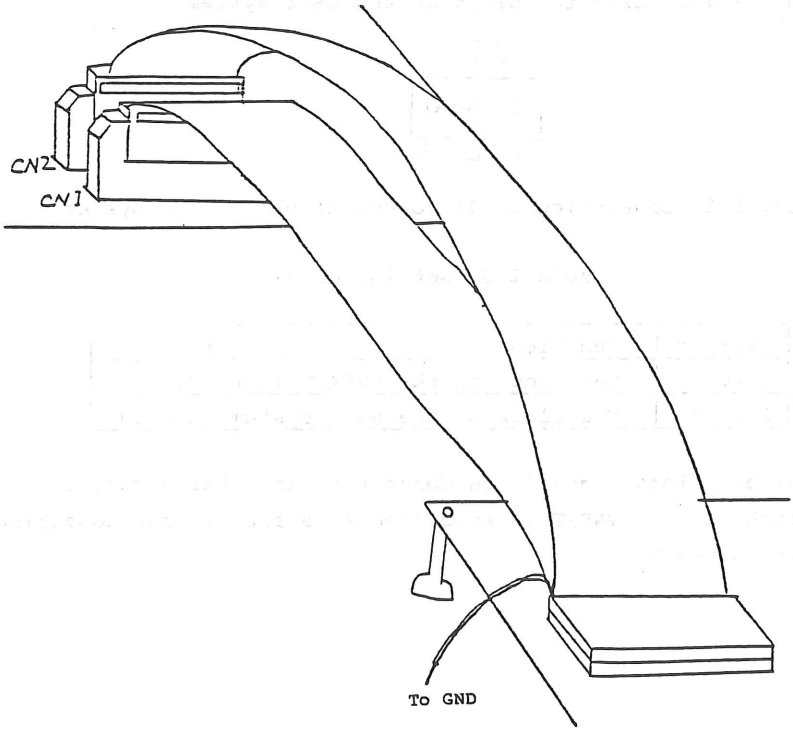
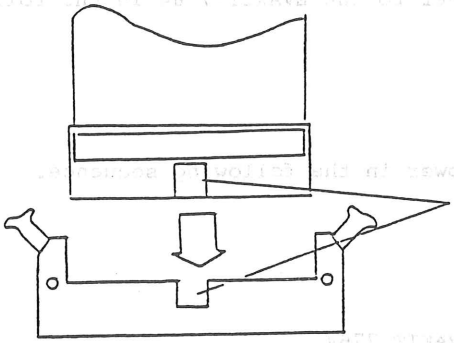


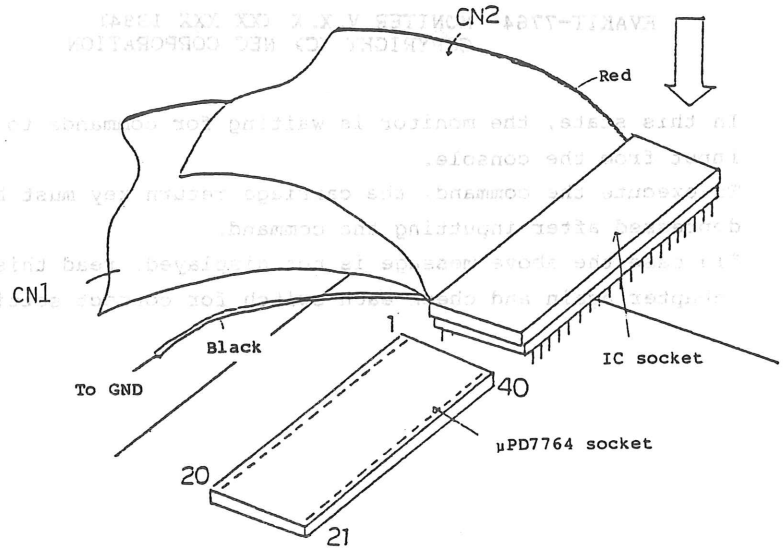
Fig. 2-5 Connection of EVAKIT-7764 to User System



Connect the cable so that it aligns with the groove on the connector.

Although both CN1 and CN2 can be connected to the cable in the same way, be sure to connect CN1 to socket CN1, and CN2 to socket CN2 of the emulation cable.

Fig. 2-6 Connection of User System Connecting Connectors (CN1 and CN2)



Connect the cable so that the red side of the cable comes to pin 1 side. Connect the GND connecting cable to the GND at the user system side.

Fig. 2-7 Connection of µPD7764 Socket on User System

2.2 Power ON/OFF Sequence

Turn on the power to the EVAKIT-7764 in the following sequence.

(1) +5V
(2) +12V
(3) -12V

Turn off the power in the following sequence.

(1) -12V
(2) +12V
(3) +5V

2.3 Starting up EVAKIT-7764

The EVAKIT-7764 system is started by application of power and the reset switch (SW1).
When the system is started up, the following message is displayed on the console, followed by prompt mark "*" that indicates that the EVAKIT is in the command wait state.

```
EVAKIT-7764  MONITER V.X.X (XX XXX 1984)
             COPYRIGHT (C) NEC CORPORATION
```

In this state, the monitor is waiting for commands to be input from the console.
To execute the command, the carriage return key must be depressed after inputting the command.
*In case the above message is not displayed, read this chapter again and check each switch for correct setting.

CHAPTER 3 MONITOR COMMANDS

3.1 Outline

The EVAKIT-7764 commands are broadly divided into the following six groups.

(1) DUMP command group : Displays the contents of each RAM and internal register.

(2) CHANGE command group : Changes the contents of each RAM and internal register.

(3) GO command group : Executes the program of the μPD7764 from the specified address to the break point.

(4) USER command group : Changes the mode between the USER and EVAKIT.

(5) SAVE command group : Outputs the contents of each RAM and internal register to the RS-232C interface.

(6) LOAD command group : Inputs data from the RS-232C interface to each RAM and internal register.

3.1.1 Notes on command description

(1) If key input is performed exceeding the specified number of digits, the later-input characters equal to the specified number of digits are valid. If the number of input digits is less than that specified, higher digits are assumed to be 0.

Ex. If the valid number of digits is 4
If "10000" is input, "0000" is assumed.
If "12" is input, "0012" is assumed.

Ex. If the valid number of digits is 2
If "101C" is input, "1C" is assumed.
If "5" is input, "05" is assumed.

(2) Both upper- and lower-case alphabetical characters may be used and no distinction is made between them.

(3) All numerals for inputting commands must be in hexadecimal format.

(4) If a wrong character is input by mistake, it can be canceled by the DEL key (08H).

(5) In case an error occurs in the inputting format of a command, the following message is displayed and the system reenters the command input wait state. If this happens, input the correct command.
"COMMAND ERROR"

(6) In case an error occurs in the inputting format of a bank or address, the following message is displayed and the system reenters the command input state. If this happens, input the correct bank or address.
"OPERAND ERROR"

(7) Each RAM of the μPD7764 is different in bank and address capacity from the others. If a value exceeding the bank or address capacity of each RAM in this system is input, the maximum value of the RAM is set.
Ex. RAM-A (bank capacity: 00H to 15H)
If "3F" is input as the value of a bank, "15" is assumed.
If "10F" is input as the vaue of a bank, "0F" is assumed.

(8) Legend

xxxx,yyyy	:	Pattern memory address (hexadecimal numbers, 4 digits max.)
xx,yy	:	Instruction memory address (hexadecimal numbers, 2 digits max.)
mm,nn	:	RAM-A bank (hexadecimal numbers, 2 digits max.)
m,n	:	RAM-B or RAM-G bank (hexadecimal numbers, 1 digit)
zz	:	RAM-G address (hexadecimal numbers, 2 digits max.)
z	:	RAM-A or RAM-B address (hexadecimal numbers, 1 digit)
LLLL	:	Break count (hexadecimal numbers, 4 digits max.)

IM1 : Instruction memory 1

IM2 : Instruction memory 2

RNAME : Each register name

PMEM : Pattern memory

↵ : Carriage return

[ ] : Indicates that the numeric value enclosed in the brackets can be omitted

*(asterisk): Prompt. Indicates that the system is in the command wait state. The prompt is printed by the system monitor.

-(hyphen) : Input request. Printed by the system monitor after inputting the CHANGE command

/(slash) : Causes the command under execution to be completed and the system to enter the command input wait state

Δ : Space (blank)

? : Printed by the system monitor when a data input error occurs during execution of the CHANGE command

3.1.2 List of commands

Table 3-1 lists the monitor commands of the EVAKIT-7764.

Table 3-1 List of Monitor Commands

Classi-fication	Command	Operation	Function
DUMP	Dl	Dl↵	Displays the contents of instruction memory 1 (IM1) from address 00 to 1F in hexadecimal format
		Dl△xx↵	Displays the contents of IM1 from address xx in hexadecimal format
		Dl△,yy↵	Displays the contents of IM1 from address 00 to yy in hexadecimal format
		Dl△xx,yy↵	Displays the contents of IM1 from address xx to yy in hexadecimal format
	DlX	DlX↵	Displays the contents of IM1 from address 00 to 1F in mnemonic format
		DlX△xx↵	Displays the contents of IM1 from address xx in mnemonic format
		DlX△,yy↵	Displays the contents of IM1 from address 00 to yy in mnemonic format
		DlX△xx,yy↵	Displays the contents of IM1 from address xx to yy in mnemonic format

Classi-fication	Command	Operation	Function
DUMP	D2	D2↵	Displays the contents of instruction memory 2 (IM2) from address 00 to 7F in hexadecimal format
		D2Δxx↵	Displays the contents of IM2 from address xx in hexadecimal format
		D2Δ,yy↵	Displays the contents of IM2 from address 00 to yy in hexadecimal format
		D2Δxx,yy↵	Displays the contents of IM2 from address xx to yy in hexadecimal format
	D2X	D2x↵	Displays the contents of IM2 from address 00 to 7F in mnemonic format
		D2XΔxx↵	Displays the contents of IM2 from address xx in mnemonic format
		D2XΔ,yy↵	Displays the contents of IM2 from address 00 to address yy in mnemonic format
		D2XΔxx,yy↵	Displays the contents of IM2 from address xx to yy in mnemonic format
	DR	DR↵	Displays the contents of all the internal registers in hexadecimal format
		DRΔRNAME↵	Displays the contents of the register specified by RNAME in hexadecimal format

Classi-fication	Command	Operation	Function
DUMP	DP	DP↵	Displays the contents of the pattern memory (PMEM) from address 0000 to FFFF in hexadecimal format
		DP△xxxx↵	Displays the contents of PMEM from address xxxx in hexadecimal format
		DP△,yyyy↵	Displays the contents of PMEM from address 0000 to yyyy in hexadecimal format
		DP△xxxx, yyyy↵	Displays the contents of PMEM from address xxxx to yyyy in hexadecimal format
CHANGE	C1	C1↵	Modifies the contents of IM1 in hexadecimal format from address 00
		C1△xx↵	Modifies the contents of IM1 in hexadecimal format from address xx
	C1X	C1X↵	Modifies the contents of instruction memory 1 (IM1) from address 00 in mnemonic format
		C1X△xx↵	Modifies the contents of IM1 from address xx in mnemonic format
	C2	C2↵	Modifies the contents of instruction memory 2 (IM2) from address 00 in hexadecimal format
		C2DWxx↵	Modifies the contents of IM2 from address xx in hexadecimal format

Classi-fication	Command	Operation	Function
CHANGE	C2X	C2X↵	Modifies the contents of IM2 from address 00 in mnemonic format
		C2XΔxx↵	Modifies the contents of IM2 from address xx in mnemonic format
	CA	CA↵	Modifies the contents of RAM-A from address 0 of bank 00 in hexadecimal format
		CAΔ,z↵	Modifies the contents of RAM-A from address z of bank 00 in hexadecimal format
		CAΔmm↵	Modifies the contents of RAM-A from address 0 of bank mm in hexadecimal format
		CAΔmm,z↵	Modifies the contents of RAM-A from address z of bank mm in hexadecimal format
	CB	CB↵	Modifies the contents of RAM-B from address 0 of bank 0 in hexadecimal format
		CBΔ,z↵	Modifies the contents of RAM-B from address z of bank 0 in hexadecimal format
		CBΔm↵	Modifies the contents of RAM-B from address 0 of bank m in hexadecimal format
		CBΔm,z↵	Modifies the contents of RAM-B from address z of bank m in hexadecimal format
	CG	CG↵	Modifies the contents of RAM-G from address 00 of bank 0 in hexadecimal format
		CGΔ,zz↵	Modifies the contents of RAM-G from address zz of bank 0 in hexadecimal format

Classi-fication	Command	Operation	Function
CHANGE		CGΔm↵	Modifies the contents of RAM-G from address 00 of bank m in hexadecimal format
		CGΔm,zz↵	Modifies the contents of RAM-G from address zz of bank m in hexadecimal format
	CR	CR↵	Modifies the contents of the internal registers from the address of the IR register in hexadecimal format
		CRΔRNAME↵	Modifies the contents of the the internal registers from the register specified by RNAME in hexadecimal format
	CP	CP↵	Modifies the contents of PMEM from address 0000 in hexadecimal format
		CPΔxxxx↵	Modifies the contents of PMEM from address xxxx in hexadecimal format
GO	GO	GO↵	Executes the program from address 00
		GOΔxx↵	Executes the program from address xx
		GOΔ,yy↵	Executes the program from address 00. The program execution continues until it passes break point (at address yy) once
		GOΔxx,yy↵	Executes the program from address xx. The program execution continues until it passes break point (at address yy) once

Classi-fication	Command	Operation	Function
GO		GOΔxx,yy, LLLL ↵	Executes the program from address nn. The program execution continues until it passes the break point (at address yy) LLLL times
USER	USR	USR↵	Connects the μPD7764 bus on the EVAKIT-7764 to the user system
SAVE	S	SΔA↵	Outputs the contents of IM1 and IM2 to the RS-232C interface
	SW	SWΔA↵	Outputs the contents of the internal registers, RAM-A, RAM-B, and RAM-G to the RS-232C interface
	SP	SPΔA,xxxx↵	Outputs the contents of PMEM from address xxxx to yyyy to the RS-232C interface
LOAD	L	LΔA↵	Inputs data from the RS-232C interface to IM1 and IM2
	L1	L1ΔA↵	Inputs data from the RS-232C interface to IM1
	L2	L2ΔA↵	Inputs data from the RS-232 interface to IM2
	LW	LWΔA↵	Inputs data from the RS-232C interface to the internal registers, RAM-A, RAM-B, and RAM-G
	LP	LP↵	Inputs data from the RS-232C interface to PMEM
		LPΔAmxxxx↵	Inputs data from the RS-232C to PMEM, starting from address xxxx

NOTE: Since there are some restrictions to using commands, refer to 3.2, Detailed command description, for details.

3.2 Detailed Command Description

3.2.1 DUMP command

(1) Dl command

```
* D1Δ[xx][,yy]↵
```

Displays the contents of instruction memory 1 (IM1) in hexadecimal format. The parameters of this command must satisfy this condition.

xx<yy≤1F

a. When the prompt (*) is displayed, input the Dl command followed by carriage return. The contents of IM1 from address 00 to 1F are then displayed in hexadecimal format. After the IM1 contents have been displayed, the prompt is displayed again and the program enters the command input wait state.
Example:

```
* D1↵

< DUMP IM1 >
00 :  E000  2800  7800  0040  9001  8000  D009  C00A
08 :  B016  A00F  0140  2800  7800  1000  0180  2800
10 :  7800  1800  2800  7800  D016  B016  6800  7000
18 :  1840  2800  7800  0000  0000  0000  0000  0000
*
```

b. When the prompt (*) is displayed, input the Dl command, a space, a desired address, and depress carriage return. The contents of the specified address of IM1 are then displayed in hexadecimal format. After the display of the IM1 contents, the prompt is displayed again and the program enters the command input wait state.
Example: To display the contents of address 15 of IM1 in hexadecimal format

```
* D1Δ15↵

< DUMP IM1 >
15 :  B016
*
```

c. When the prompt (*) is displayed, input the D1 command, a comma, a desired address, and depress carriage return. The contents of IM1 from address 00 to the specified address are then displayed in hexadecimal format. After the display of the IM1 contents, the prompt is displayed again, and the program enters the command input wait state.
Example: To display the contents of IM1 from address 00 to 12 in hexadecimal format

```
* D1△,12↵

< DUMP IM1 >
00 :  E000  2800  7800  0040  9001  8000  D009  C00A
08 :  B016  A00F  0140  2800  7800  1000  0180  2800
10 :  7800  1800  2800
*
```

d. When the prompt (*) is displayed, input the D1 command, a comma, a desired start address, a comma, and a desired end address, and depress carriage return. The contents of IM1 from the specified start address to end address are then displayed in hexadecimal format. After the display of the IM1 contents, the prompt is displayed again, and the program enters the command input wait state.
Example: To display the contents of IM1 from address 10 to 1C in hexadecimal format

```
* D1△10,1C↵

< DUMP IM1 >
10 :  7800  1800  2800  7800  D016  B016  6800  7000
18 :  1840  2800  7800  0000  0000
*
```

(2) D1X command

```
* D1XΔ [xx] [,yy]↵
```

Displays the contents of instruction memory 1 (IM1) in mnemonic format. The parameters of this command must satisfy this condition.
xx<yy≤1F

a. When the prompt (*) is displayed, input the D1X command followed by carriage return. The contents of IM1 from address 00 to 1F are then displayed in mnemonic format. After the display of the IM1 contents, the prompt is displayed again and the program enters the command input wait state. However, when the contents are displayed to the maximum capacity of the display screen (i.e., 21 lines), the program enters the command input wait state. To continue the display to the next screen, input carriage return; otherwise, input a slash (/) and depress carriage return.

Example:

```
* D1X↵

< DUMP IM1 >
00 :  E000  WCNTA,000
01 :  2800  SDF  ,NOP ,NOP ,NOP ,NOP ,NOP
02 :  7800  WAIT ,NOP ,NOP ,NOP ,NOP ,NOP
03 :  0040  NOP  ,NOP ,WB1 ,NOP ,NOP ,NOP
04 :  9001  PC1  ,001
05 :  8000  RCNT ,000
06 :  D009  BARH ,009
07 :  C00A  BARL ,00A
08 :  B016  LCNTH,016
09 :  A00F  LCNTL,00F
0A :  0140  NOP  ,NOP ,RI  ,NOP ,NOP ,NOP
0B :  2800  SDF  ,NOP ,NOP ,NOP ,NOP ,NOP
0C :  7800  WAIT ,NOP ,NOP ,NOP ,NOP ,NOP
0D :  1000  DLJ  ,NOP ,NOP ,NOP ,NOP ,NOP
0E :  0180  NOP  ,NOP ,RC1 ,NOP ,NOP ,NOP
0F :  2800  SDF  ,NOP ,NOP ,NOP ,NOP ,NOP
10 :  7800  WAIT ,NOP ,NOP ,NOP ,NOP ,NOP
11 :  1800  DHJ  ,NOP ,NOP ,NOP ,NOP ,NOP
12 :  2800  SDF  ,NOP ,NOP ,NOP ,NOP ,NOP
13 :  7800  WAIT ,NOP ,NOP ,NOP ,NOP ,NOP
14 :  D016  BARH ,016
↵ <-------------------------------------------

15 :  B016  LCNTH,016
16 :  6800  SIOB ,NOP ,NOP ,NOP ,NOP ,NOP
17 :  7000  WIOB ,NOP ,NOP ,NOP ,NOP ,NOP
18 :  1840  DHJ  ,NOP ,WB1 ,NOP ,NOP ,NOP
19 :  2800  SDF  ,NOP ,NOP ,NOP ,NOP ,NOP
1A :  7800  WAIT ,NOP ,NOP ,NOP ,NOP ,NOP
1B :  0000  NOP  ,NOP ,NOP ,NOP ,NOP ,NOP
1C :  0000  NOP  ,NOP ,NOP ,NOP ,NOP ,NOP
1D :  0000  NOP  ,NOP ,NOP ,NOP ,NOP ,NOP
1E :  0000  NOP  ,NOP ,NOP ,NOP ,NOP ,NOP
1F :  0000  NOP  ,NOP ,NOP ,NOP ,NOP ,NOP
*
```

A maximum of 21 lines can be displayed and the system waits for a command to be input. Depressing carriage return under this condition causes the next screen to be displayed.

b. When the prompt (*) is displayed, input the DlX command, a space, a desired address, and depress carriage return. The contents of the specified address of IMl are then displayed in mnemonic format. After the display of IMl contents, the prompt is displayed again and the program enters the command input state.

Example: To display the contents of address 11 of IMl in mnemonic code

```
* DlX△11↵

< DUMP IMl >
11 :  1800  DHJ  ,NOP ,NOP ,NOP ,NOP ,NOP
*
```

c. When the prompt (*) is displayed, input the DIX command, a space, a comma, a desired address, and depress carriage return. The contents of IMl from address 00 to the specified address are then displayed in mnemonic format. After the display of the IMl contents, the prompt is displayed again and the program enters the command input wait state. However, if the contents of IMl are displayed to the maximum capacity of the display screen (i.e., 21 lines), the program enters the command input wait state. To continue the display to the next screen, depress carriage return; otherwise, input a slash and depress carriage return.

Example: To display the contents of IMl from address 00 to 05 in mnemonic format

```
* DlX△,5↵

< DUMP IMl >
00 :  E000  WCNTA,000
01 :  2800  SDF  ,NOP ,NOP ,NOP ,NOP ,NOP
02 :  7800  WAIT ,NOP ,NOP ,NOP ,NOP ,NOP
03 :  0040  NOP  ,NOP ,WBl ,NOP ,NOP ,NOP
04 :  9001  PCl  ,001
05 :  8000  RCNT ,000
*
```

d. When the prompt (*) is displayed, input the DlX command, a space, a desired start address, a comma, a desired end address, and depress carriage return. The contents of IMl from the specified start address to the end address are then displayed in mnemonic format. After the display of the IMl contents, the prompt is displayed again and the program enters the command input wait state. However, if the contents of IMl are displayed to the maximum capacity of the display screen (i.e., 21 lines) the program enters the command input wait state. To continue the display to the next screen, input carriage return; otherwise, input a slash and depress carriage return.

Example: To display the contents of IMl from address 14 to 1C

```
* DlX△14,16↵

< DUMP IMl >
14 :  D016  BARH ,016
15 :  B016  LCNTH,016
16 :  6800  SIOB ,NOP ,NOP ,NOP ,NOP ,NOP
*
```

(3) D2 command

```
* D2Δ [xx] [,yy]↵
```

Displays the contents of instruction memory 2 (IM2) in hexadecimal format. The parameters of this command must satisfy this condition.

xx<yy≤7F

a. When the prompt (*) is displayed, input the D2 command followed by carriage return. The contents of IM2 from address 00 to 7F are then displayed in hexadecimal format. After the display of the IM2 contents, the prompt is displayed again and the program enters the command input wait state.

Example:

```
* D2↵

< DUMP IM2 >
00 : 001C 0018 6800 0170 915E D805 5200 0016
08 : 5400 0010 904C D60B 0028 003A C80A 800E
10 : D810 0039 C808 0018 914C 6A00 01D0 4A00
18 : 0005 D819 8009 800E 0018 5200 0016 5400
20 : 0010 D821 88BC D623 800E 5A00 00FF 1B2B
28 : 1B28 1798 CA2C 002C 0028 003A C821 0039
30 : C81F 0018 D832 915C 9040 D635 6800 1070
38 : 800E D839 905C 9820 DE3C C03D 0000 0000
40 : 0000 0000 0000 0000 0000 0000 0000 0000
48 : 0000 0000 0000 0000 0000 0000 0000 0000
50 : 0000 0000 0000 0000 0000 0000 0000 0000
58 : 0000 0000 0000 0000 0000 0000 0000 0000
60 : 0000 0000 0000 0000 0000 0000 0000 0000
68 : 0000 0000 0000 0000 0000 0000 0000 0000
70 : 0000 0000 0000 0000 0000 0000 0000 0000
78 : 0000 0000 0000 0000 0000 0000 0000 0000
*
```

b. When the prompt (*) is displayed, input the D2 command, a space, a desired address, and depress carriage return. The contents of the specified address of IM2 are then displayed in hexadecimal format. After the display of the IM2 contents, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of address 0B of IM2 in hexadecimal format

```
* D2△B↵

< DUMP IM2 >
0B : D60B
*
```

c. When the prompt (*) is displayed, input the D2 command, a space, a comma, a desired address, and depress carriage return. The contents of IM2 from address 00 to the specified address are then displayed in hexadecimal format. After the display of the IM2 contents, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of IM2 from address 00 to 15 in hexadecimal format

```
* D2△,15↵

< DUMP IM2 >
00 : 001C  0018  6800  0170  915E  D805  5200  0016
08 : 5400  0010  904C  D60B  0028  003A  C80A  800E
10 : D810  0039  C808  0018  914C  6A00
*
```

d. When the prompt (*) is displayed, input the D2 command, a space, a comma, a desired start and end addresses, and depress carriage return. The contents of IM2 from the specified start address to end address are then displayed in hexadecimal format. After the display of the IM2 contents, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of IM2 from address 19 to 1E in hexadecimal format

```
* D2△19,1E↵

< DUMP IM2 >
18 : 0005  D819  8009  800E  0018  5200  0016
*
```

(4) D2X command

```
* D2XΔ [xx] [,yy]↵
```

Displays the contents of instruction memory 2 (IM2) in mnemonic format. The parameters of this command must satisfy this condition.

xx<yy≤7F

a. When the prompt (*) is displayed, input the D2X command followed by carriage return. The contents of IM2 from address 00 to 7F are then displayed in mnemonic format. After the display of the IM2 contents, the prompt is displayed again and the program enters the command input wait state. When the contents of IM2 are displayed to the maximum capacity of the display screen (i.e., 21 lines), however, the program enters the command input wait state. To continue the display to the next screen, depress carriage return; otherwise, input a slash and depress carriage return.

Example:

```
* D2X↵

< DUMP IM2 >
00 :  001C          CLR     GR4     ,NON
01 :  0018          CLR     GR0     ,NON
02 :  6800  0170    NOP     NON     ,IOSR1  ,0170
04 :  915E          MOV     IOSR2   ,GR0    ,STTD
05 :  D805          JMP     NDF     ,05
06 :  5200  0016    NOP     NON     ,GR1    ,0016
08 :  5400  0010    NOP     NON     ,GR2    ,0010
0A :  904C          MOV     IR      ,GR0    ,RDF
0B :  D60B          JMP     IOB     ,0B
0C :  0028          INC     GR0     ,NON
0D :  003A          DEC     GR2     ,NON
0E :  C80A          JMP     NZ      ,0A
0F :  800E          MOV     NON     ,NON    ,STTD
10 :  D810          JMP     NDF     ,10
11 :  0039          DEC     GR1     ,NON
12 :  C808          JMP     NZ      ,08
13 :  0018          CLR     GR0     ,NON
14 :  914C          MOV     IOSR1   ,GR0    ,RDF
15 :  6A00  01D0    NOP     NON     ,IOSR2  ,01D0
17 :  4A00  0005    NOP     NON     ,PC1    ,0005
19 :  D819          JMP     NDF     ,19
/↵
*
```

← When the display continues to the maximum capacity of the screen, the program enters the command input wait state. Depress carriage return to continue the display to the next screen.

```
1A :  8009          MOV     NON     ,NON    ,SIOB
1B :  800E          MOV     NON     ,NON    ,STTD
1C :  0018          CLR     GR0     ,NON
1D :  5200  0016    NOP     NON     ,GR1    ,0016
1F :  5400  0010    NOP     NON     ,GR2    ,0010
21 :  D821          JMP     NDF     ,21
22 :  88BC          MOV     GR3     ,IR     ,RDF
23 :  D623          JMP     IOB     ,23
24 :  800E          MOV     NON     ,NON    ,STTD
25 :  5A00  00FF    NOP     NON     ,GR5    ,00FF
27 :  1B2B          AND     GR3     ,GR5
28 :  1B28          AND     GR0     ,GR5
29 :  1798          CMPR    GR0     ,GR3
2A :  CA2C          JMP     Z       ,2C
2B :  002C          INC     GR4     ,NON
2C :  0028          INC     GR0     ,NON
2D :  003A          DEC     GR2     ,NON
2E :  C821          JMP     NZ      ,21
2F :  0039          DEC     GR1     ,NON
30 :  C81F          JMP     NZ      ,1F
31 :  0018          CLR     GR0     ,NON
/↵
*
```

← When the display continues to the maximum capacity of the screen, the program enters the command input wait state. Input a slash and depress carriage return to allow the program to stay in the current state.

b. When the prompt (*) is displayed, input the D2X command, a space, a desired address, and depress carriage return. The contents of the specified address of IM2 are then displayed in mnemonic format. After the display of the IM2 contents, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of address 35 of IM2 in mnemonic format

```
* D2X△35↵

< DUMP IM2 >
35 :  D635          JMP    IOB    ,35
*
```

c. When the prompt (*) is displayed, input the D2X command, a space, a comma, a desired address, and depress carriage return. The contents of IM2 from address 00 to the specified address are then displayed in mnemonic format. After the display of the IM2 contents, the prompt is displayed again and the program enters the command input wait state. However, when the contents of IM2 are displayed to the maximum capacity of the display screen (i.e., 21 lines), the program enters the command input wait state. To continue the display to the next screen, depress carriage return; otherwise, input a slash followed by carriage return.

Example: To display the contents of IM2 from address 00 to 0A in mnemonic format

```
* D2X△,A↵

< DUMP IM2 >
00 :  001C          CLR    GR4    ,NON
01 :  0018          CLR    GR0    ,NON
02 :  6800    0170  NOP    NON    ,IOSR1  ,0170
04 :  915E          MOV    IOSR2  ,GR0    ,STTD
05 :  D805          JMP    NDF    ,05
06 :  5200    0016  NOP    NON    ,GR1    ,0016
08 :  5400    0010  NOP    NON    ,GR2    ,0010
0A :  904C          MOV    IR     ,GR0    ,RDF
*
```

d. When the prompt (*) is displayed, input the D2X command, a space, desired start and end addresses with a comma inserted to delimit them, and depress carriage return. The contents of IM2 from the specified start address to end address are displayed in mnemonic format. After the display of the IM2 contents, the prompt is displayed again and the program enters the command input wait state. However, when the contents of IM2 are displayed to the maximum capacity of the display screen (i.e., 21 lines), the program enters the command input wait state again. To continue the display to the next screen, depress carriage return; otherwise, input a slash followed by carriage return.

Example: To display the contents of IM2 from address 1F to 32 in mnemonic format

```
* D2X△1F,32↵

< DUMP IM2 >
1F :  5400   0010  NOP   NON   ,GR2   ,0010
21 :  D821         JMP   NDF   ,21
22 :  88BC         MOV   GR3   ,IR    ,RDF
23 :  D623         JMP   IOB   ,23
24 :  800E         MOV   NON   ,NON   ,STTD
25 :  5A00   00FF  NOP   NON   ,GR5   ,00FF
27 :  1B2B         AND   GR3   ,GR5
28 :  1B28         AND   GR0   ,GR5
29 :  1798         CMPR  GR0   ,GR3
2A :  CA2C         JMP   Z     ,2C
2B :  002C         INC   GR4   ,NON
2C :  0028         INC   GR0   ,NON
2D :  003A         DEC   GR2   ,NON
2E :  C821         JMP   NZ    ,21
2F :  0039         DEC   GR1   ,NON
30 :  C81F         JMP   NZ    ,1F
31 :  0018         CLR   GR0   ,NON
32 :  D832         JMP   NDF   ,32
*
```

(5) DA command

```
* DA␣[mm] [, nn]↵
```

Displays the contents of RAM-A in hexadecimal format.

The parameters of this command must satisfy this condition.

mm<nn<15

a. When the prompt (*) is displayed, input the DA command and depress carriage return. The contents of RAM-A from bank 00 to 15 are displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example:

```
* DA↵
< DUMP RAM-A >
00 : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
01 : 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
02 : 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
03 : 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
04 : 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
05 : 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
06 : 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
07 : 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
08 : 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
09 : 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
0A : A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
0B : B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
0C : C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
0D : D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
0E : E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
0F : F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
10 : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
11 : 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
12 : 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
13 : 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
14 : 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
15 : 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
*
```

b. When the prompt (*) is displayed, input the DA command, a space, a desired address, and depress carriage return. The contents of the specified bank of RAM-A are then displayed in hexadecimal format. After the display, the prompt is displayed again, and the program enters the command input wait state.

Example: To display the contents of bank 15 of RAM-A in hexadecimal format

```
* DA␣15↵
< DUMP RAM-A >
15 : 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
*
```

c. When the prompt (*) is displayed, input the DA command, a space, a comma, a desired address, and depress carriage return. The contents of RAM-A from bank 00 to the specified bank are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of RAM-A from bank 00 to 02 in hexadecimal code

```
* DA␣,2 ↵

< DUMP RAM-A >
00 :  00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
01 :  10  11  12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F
02 :  20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
*
```

d. When the prompt (*) is displayed, input the DA command, a space, a desired start bank, a comma, a desired end bank, and depress carriage return. The contents of RAM-A from the specified start bank to end bank are then displayed. After the display, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of RAM-A from bank 10 to 15 in hexadecimal format

```
* DA␣10,15 ↵

< DUMP RAM-A >
10 :  00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
11 :  10  11  12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F
12 :  20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
13 :  30  31  32  33  34  35  36  37  38  39  3A  3B  3C  3D  3E  3F
14 :  40  41  42  43  44  45  46  47  48  49  4A  4B  4C  4D  4E  4F
15 :  50  51  52  53  54  55  56  57  58  59  5A  5B  5C  5D  5E  5F
*
```

(6) DB command

```
* DB△[m] [,n]↵
```

Displays the contents of RAM-B in hexadecimal format.

The parameters of this command must satisfy this condition.

m<n≤3

a. When the prompt (*) is displayed, input the DB command followed by carriage return. The contents of RAM-B from bank 0 to 3 are displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example:

```
* DB↵
< DUMP RAM-B >
00 :  00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
01 :  10  11  12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F
02 :  20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
03 :  30  31  32  33  34  35  36  37  38  39  3A  3B  3C  3D  3E  3F
*
```

b. When the prompt (*) is displayed, input the DB command, a space, a desired bank, and depress carriage return. The contents of RAM-B at the specified bank are displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of bank 2 of RAM-B in hexadecimal format

```
* DB△2↵
< DUMP RAM-B >
02 :  20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
*
```

c. When the prompt (*) is displayed, input the DB command, a space, a comma, and a desired bank, and then depress carriage return. The contents of RAM-B from bank 0 to the specified bank are then displayed. After the display, the prompt is displayed again and the program enters the command input wait state.
Example: To display the contents of RAM-B from bank 0 to 2 in hexadecimal format

```
* DB␣,2↵

< DUMP RAM-B >
00 :  00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
01 :  10  11  12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F
02 :  20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
*
```

d. When the prompt (*) is displayed, input the DB command, a space, a desired start bank, a comma, an end bank, and then depress carriage return. The contents of RAM-B from the specified start bank to end bank are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.
Example: To display the contents of RAM-B from bank 2 to 3 in hexadecimal format

```
* DB␣2,3↵

< DUMP RAM-B >
02 :  20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
03 :  30  31  32  33  34  35  36  37  38  39  3A  3B  3C  3D  3E  3F
*
```

(7) DG command

```
* DG␣ [m] [, n] ↵
```

Displays the contents of RAM-G in hexadecimal format.
The parameters of this command must satisfy this condition.
$m < n \leq 3$

a. When the prompt (*) is displayed, input the DG command followed by carriage return. The contents of RAM-G from bank 0 to 3 are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.
Example:

```
* DG↵

< DUMP RAM-G >
[ BANK-0 ]
          00 :  0000  0001  0002  0003  0004  0005  0006  0007
          08 :  0008  0009  000A  000B  000C  000D  000E  000F
          10 :  0010  0011  0012  0013  0014  0015
[ BANK-1 ]
          00 :  1000  1001  1002  1003  1004  1005  1006  1007
          08 :  1008  1009  100A  100B  100C  100D  100E  100F
          10 :  1010  1011  1012  1013  1014  1015
[ BANK-2 ]
          00 :  2000  2001  2002  2003  2004  2005  2006  2007
          08 :  2008  2009  200A  200B  200C  200D  200E  200F
          10 :  2010  2011  2012  2013  2014  2015
[ BANK-3 ]
          00 :  3000  3001  3002  3003  3004  3005  3006  3007
          08 :  3008  3009  300A  300B  300C  300D  300E  300F
          10 :  3010  3011  3012  3013  3014  3015

*
```

b. When the prompt (*) is displayed, input the DG command, a space, and a desired bank followed by carriage return. The contents of the specified bank of RAM-G are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.
Example: To display the contents of bank 1 of RAM-G in hexadecimal format

```
* DG␣1↵

< DUMP RAM-G >
[ BANK-1 ]
          00 :  1000  1001  1002  1003  1004  1005  1006  1007
          08 :  1008  1009  100A  100B  100C  100D  100E  100F
          10 :  1010  1011  1012  1013  1014  1015

*
```

c. When the prompt (*) is displayed, input the DG command, a space, a comma, and a desired bank followed by carriage return. The contents of RAM-G from bank 0 to the specified bank are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of RAM-G from bank 0 to 2 in hexadecimal format

```
* DG␣,1↵

< DUMP RAM-G >
[ BANK-0 ]
          00 :  0000  0001  0002  0003  0004  0005  0006  0007
          08 :  0008  0009  000A  000B  000C  000D  000E  000F
          10 :  0010  0011  0012  0013  0014  0015
[ BANK-1 ]
          00 :  1000  1001  1002  1003  1004  1005  1006  1007
          08 :  1008  1009  100A  100B  100C  100D  100E  100F
          10 :  1010  1011  1012  1013  1014  1015

*
```

d. When the prompt (*) is displayed, input the DG command, a space, a desired start bank, a comma, and a desired end bank followed by carriage return. The contents of RAM-G from the specified start bank to end bank are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of RAM-G from bank 1 to 2 in hexadecimal format

```
* DG␣1,2↵

< DUMP RAM-G >
[ BANK-1 ]
          00 :  1000  1001  1002  1003  1004  1005  1006  1007
          08 :  1008  1009  100A  100B  100C  100D  100E  100F
          10 :  1010  1011  1012  1013  1014  1015
[ BANK-2 ]
          00 :  2000  2001  2002  2003  2004  2005  2006  2007
          08 :  2008  2009  200A  200B  200C  200D  200E  200F
          10 :  2010  2011  2012  2013  2014  2015

*
```

(8) DR command

```
* DR△[RNAME]↵
```

Displays the contents of the internal registers in hexadecimal format

a. When the prompt (*) is displayed, input the DR command followed by carriage return. The contents of all the internal registers are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example:

```
* DR↵

< REGISTER DUMP >
IR    - 0110  PC1   - 002D  GR0   - 0010  GR1   - 0015  GR2   - 0000
GR3   - 000F  GR4   - 0000  GR5   - 00FF  GR6   - 0009  GR7   - 0000
GARG  - 000B  MC    - 0000  DRG0  - 000D  DRG1  - 000E  IOSR1 - 0000
IOSR2 - 01D0  PCNTA - 0011  PCNTB - 0012  PC2F  - 00D3  BAR   - 090A
LCNT  - 150E  RCNT  - 0012  WCNTA - 0000  WCNTB - 0020
*
```

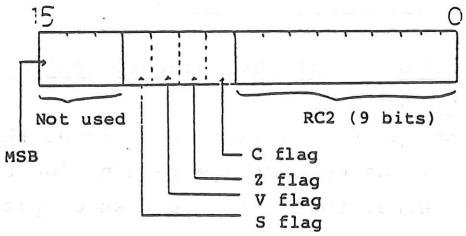
b. When the prompt (*) is displayed, input the DR command, a space, and a desired register name followed by carriage return. The contents of the specified register are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of the RCNT register in hexadecimal format

```
* DR△RCNT↵

< REGISTER DUMP >
RCNT  - 0012
*
```

*The bit configuration of the PC2F register is as follows:



NOTE: After execution of the GO command, the contents of some registers may become undefined when the DR command is executed. (For details, refer to the description of the GO command.)

(9) DP command

```
* DP△ [xxxx] [yyyy]↵
```

Displays the contents of the pattern memory (PMEM) in hexadecimal format. The parameters of this command must satisfy this condition.

xxxx<yyyy<FFFF

a. When the prompt (*) is displayed, input the DP command followed by carriage return. The contents of PMEM from address 0000 to FFFF are displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.
However, when the contents of PMEM are displayed to the maximum capacity of the display screen (i.e., 150 bytes), the program enters the command input wait state. To continue the display to the next screen, depress carriage return; otherwise, input a slash followed by carriage return.

Example:

```
* DP↵

< DUMP PATTERN-M >
0000 :  00  00  34  0F  12  00  00  0C  0E  00  00  0A  00  00  00  00
0010 :  00  03  00  0F  45  45  45  AA  45  04  0D  45  03  45  45  0E
0020 :  0D  45  45  45  0D  45  45  45  45  45  45  45  45  45  03  45
0030 :  0D  45  45  45  45  0D  45  03  45  45  0A  45  45  AC  45  03
0040 :  0E  45  04  45  0C  45  45  45  02  45  0C  45  03  45  FC  02
0050 :  45  45  45  0D  45  45  45  45  45  45  45  45  0A  01  45  45
0060 :  45  45  45  45  45  45  45  45  45  45  DE  E3  45  45  04  45
0070 :  45  45  45  FF  0F  45  45  CD  45  0D  45  0A  AD  45  45  45
0080 :  45  0D  0E  0F  0C  45  12  45  45  34  45  0F  45  45  0C  45
0090 :  DF  BB  45  5B  45  35  45  98  05  45  45  45  03  45  45  45
00A0 :  13  13  13  34  13  13  05  13  13  0D  0B  13  13  13  23  13
00B0 :  13  02  13  13  D4  0F  13  13  13  13  13  32  13  13  12  13
00C0 :  13  13  13  34  13  13  13  0B  13  13  13  13  13  13  13  23
00D0 :  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13
00E0 :  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13  13
00F0 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0100 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0110 :  0F  34  34  34  34  34  34  34  34  34  34  34  EF  34  34  34  34
0120 :  34  34  34  34  34  34  34  0F  34  0E  34  34  34  34  34  34
0130 :  34  34  34  34  34  34  34  34  34  34  34  0B  0F  34  34  34
0140 :  0E  34  0F  FB  BF  34  34  34  34  01  34  34  34  02  34  34
```

↵ ← : When the contents of PMEM are displayed to the maximum capacity of the screen, the program enters the command input wait state. Depress carriage return to continue the display to the next screen.

```
0150 :  34  ED  34  34  34  34  34  CB  34  34  34  CB  34  34  34  34
0160 :  FB  34  34  0C  34  34  34  02  20  34  34  34  34  34  34  34
0170 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0180 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0190 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
01A0 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
01B0 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
01C0 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
01D0 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
01E0 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
01F0 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0200 :  0D  34  04  34  0F  34  34  0D  02  34  34  34  34  34  34  0B
0210 :  34  34  34  0A  34  34  34  0D  34  03  34  34  34  34  34  34
0220 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0230 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0240 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0250 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0260 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0270 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0280 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
0290 :  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34  34
```

/↵ ← : When the contents of PMEM are displayed to the maximum capacity of the screen, the program enters the command input wait state. To continue the display, depress carriage return.

b. When the prompt (*) is displayed, input the DP command, a space, and a desired address followed by carriage return. The contents of PMEM at the specified address are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

Example: To display the contents of address 45DA of PMEM in hexadecimal format

```
* DP△45DA↵

< DUMP PATTERN-M >
45DA :  AC
*
```

c. When the prompt (*) is displayed, input the DP command, a space, a comma, and a desired address followed by carriage return. The contents of PMEM from address 0000 to the specified address are displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state.

However, when the contents of PMEM are displayed to the maximum capacity of the display screen (i.e., 150 bytes), the program enters the command input wait state. To continue the display to the next screen, depress carriage return; otherwise, input a slash followed by carriage return.

Example: To display the contents of PMEM from address 0000 to 001A in hexadecimal format

```
* DP△,001A↵

< DUMP PATTERN-M >
0000 :  00  00  34  0F  12  00  00  0C  0E  00  00  0A  00  00  00  00
0010 :  00  03  00  0F  45  45  45  AA  45  04  0D
*
```

d. When the prompt (*) is displayed, input the DP command, a space, a comma, and desired start and end addresses followed by carriage return. The contents of PMEM from the specified start address to end address are then displayed in hexadecimal format. After the display, the prompt is displayed again and the program enters the command input wait state. However, when the contents of PMEM are displayed to the maximum capacity of the display screen (i.e., 150 bytes), the program enters the command input wait state. To continue the display to the next screen, depress carriage return; otherwise, input a slash and depress carriage return.

Example: To display the contents of PMEM from address D400 to D4FF in hexadecimal format

```
* DP␣D400,D4FF ↵

< DUMP PATTERN-M >
D400 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D410 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D420 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D430 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D440 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D450 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D460 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D470 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D480 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D490 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D4A0 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D4B0 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D4C0 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D4D0 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D4E0 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
D4F0 : AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC AC
*
```

3.3.2 CHANGE command

When the CHANGE command (any of the commands described in this section) is input, the address specified by the command and data at that address are first displayed. The program then displays a hyphen (-) and enters the command input wait state. When the hyphen is displayed, input the data into which the current data is to be changed. When the data for the change is input, the address next to that already displayed and data at that address are displayed. Subsequently, a hyphen is displayed again, indicating that the program enters the command input wait state. When the hyphen is displayed, depressing carriage return causes the data at the currently displayed address to be unchanged and the next address to be displayed. If a colon (:) is input followed by carriage return, the displayed address is decremented by one and therefore, the preceding address is displayed. If an address is specified after the colon, the specified address will be displayed. To complete execution of the CHANGE command and return the program to the command input wait state, input a slash followed by carriage return.

(1) C1 command

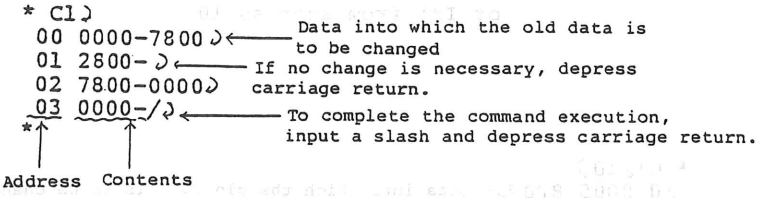
```
* C1 Δ [xx] ↵
```

Displays the contents of instruction memory 1 (IM1) in hexadecimal format. The parameters of this command must satisfy this condition.

xx≤1F

a. When the prompt (*) is displayed, input the C1 command followed by carriage return. The contents of IM1 from address 00 are then displayed in hexadecimal format. After the addresses and data of IM1 have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then, input in hexadecimal format the data into which the old data is to be changed. To change again the data already input for change, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash followed by carriage return. The prompt is then displayed and the program enters the command input wait state.

Example:



b. When the prompt (*) is displayed, input the C1 command, a space, and a desired address followed by carriage return. The contents of IM1 from the specified address are then displayed in hexadecimal format as shown in the following program example. After the display of the addresses and data of IM1, a hyphen is displayed, indicating that the program is in the command input wait state. Then, input in hexadecimal format the data into which the old data is to be changed.

To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash followed by carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in hexadecimal format the contents of IM1 from address 10

```
* C1△10↵
 10 8005-8007↵ ← Data into which the old data is to be changed
 11 A004-↵ ←—— If no change is necessary, depress carriage return.
 12 B003-:16↵ ← To display the contents of address 16
 16 F007-F010↵
 17 7800-/↵ ←—— To complete the command execution, input a slash and
*                depress carriage return.
 ↑    ↑
Address  Contents
```

(2) C1X command

```
* C1XΔ [xx]↵
```

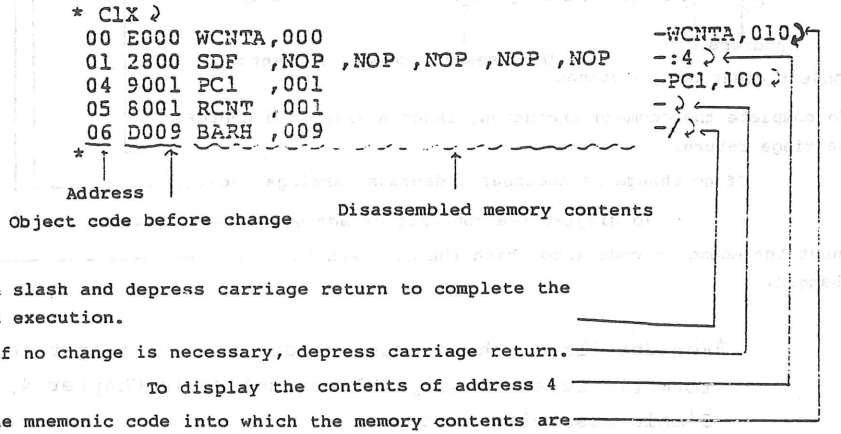
Changes in mnemonic format the contents of instruction memory 1 (IM1). The parameters of this command must satisfy this condition.

xx≤1F

a. When the prompt (*) is displayed, input the C1X command followed by carriage return. The contents of IM1 can then be changed in mnemonic format, as shown in the program example below, starting from address 00. When the command is executed, the addresses of IM1, the object codes at those addresses, and the disassembled contents of the memory are displayed. Following them, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the mnemonic code into which the contents of IM1 are to be changed. To change again the mnemonic code already input for change, input the mnemonic code for the newest change.

After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example:

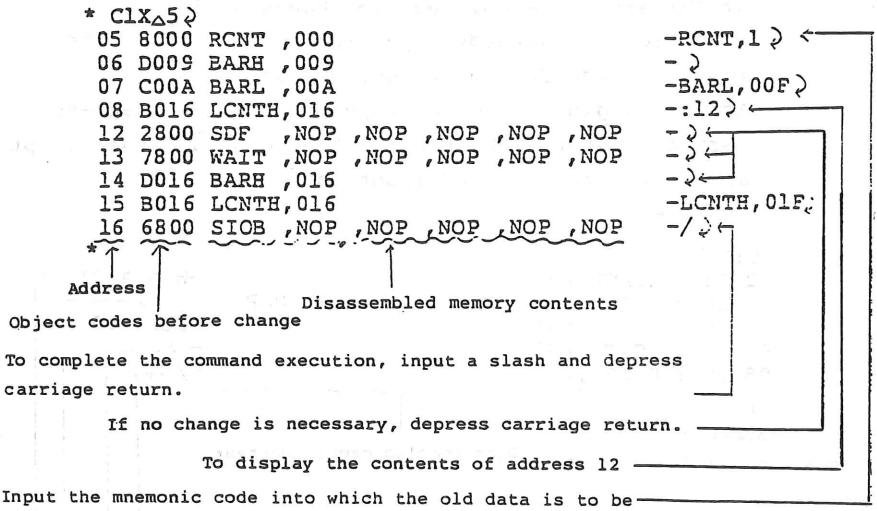


b. When the prompt (*) is displayed, input the ClX command, a space, and a desired address followed by carriage return. The contents of IM1 from the specified address are then displayed as shown in the following program example. When this command is executed, the addresses, object codes, and the disassembled contents of the memory are displayed followed by a hyphen that indicates that the program is in the command input wait state. Then input the mnemonic code into which the old data is to be changed.

To change again the data already input, input the mnemonic code for the newest change.

After changing the data, input a slash followed by depression of carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in mnemonic format the contents of IM1 from address 05



*For details on the mnemonic codes for IM1 (inputting format), refer to 4.3, IM1 assembler, in Chapter 4, Simple assembler syntax.

(3) C2 command

```
* C2Δ [xx]↵
```

Changes in hexadecimal format the contents of IM2. The parameters of this command must satisfy this condition.

xx< 7F

a. When the prompt (*) is displayed, input the C2 command followed by carriage return. The contents of IM2 from address 00 are then displayed as shown in the following program example. After the addresses and data of IM2 have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 4 digits of hexadecimal numbers.
To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing data, input a slash followed by carriage return. The program then displays the prompt and enters the command input wait state.
Example:

```
* C2↵
 00 001C-001A↵ <---- Input the data into which the old data is
 01 0018-001B↵       to be changed.
 02 6800-:C↵ <------ To display the contents of address C
 0C 0028-002A↵
 0D 003A-↵ <-------- If no change is necessary, depress carriage
 0E C80A-C80C↵       return.
 0F 800E-/↵ <------- To complete the command execution, input a
*                    slash and depress carriage return.
 ↑  ↑
Address  Contents
```

b. When the prompt (*) is displayed, input the C2 command, a space, and a desired address followed by carriage return. The contents of IM2 from the specified address are then displayed as shown in the following program example. After the addresses and data of IM2 have been displayed, a hyphen is displayed indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 4 digits of hexadecimal numbers.
To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash followed by carriage return. The program then displays the prompt and enters the command input wait state.
Example: To change in hexadecimal format the contents of IM2 from address 20

```
* C2␣20↵
 20 0010-0020↵ ←—— Input the data into which the old data is
 21 D821-:26↵  ←—— to be changed.
 26 00FF-0100↵     To display the contents of address 26
 27 1B2B-:2E↵
 2E C821-C829↵
 2F 0039-↵     ←—— If no change is necessary, depress carriage
 30 C81F-C829↵     return.
 31 0018-/↵    ←—— To complete the command execution, input a
*                  slash and depress carriage return.
 ↑  ↑
Address  Contents
```

(4) C2X command

```
* C2XΔ [xx]↵
```

Changes in mnemonic format the contents of IM2. The parameters of this command must satisfy this condition.

xx≤7F

a. When the prompt (*) is displayed, input the C2X command followed by carriage return. The contents of IM2 from address 00 are then displayed as shown in the following program example. After the addresses, the object codes, and the disassembled contents of the memory have been displayed, a hyphen is displayed indicating that the program is in the command input wait state. Then input the mnemonic code into which the old data is to be changed.

To change again the data already input, input the mnemonic code for the newest change.

After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example:

```
* C2X↵
 00 001C        CLR   GR4   ,NON              -CLR GR1,NON↵
 01 0018        CLR   GR0   ,NON              -CLR GR2,NON↵
 02 6800  0170  NOP   NON   ,IOSR1  ,0170     -:6↵
 06 5200  0016  NOP   NON   ,GR1    ,0016     -MVI GR3,0016↵
 08 5400  0010  NOP   NON   ,GR2    ,0010     -MVI GR4,0010↵
 0A 904C        MOV   IR    ,GR0    ,RDF      -/↵
 *
```

Address
Object code
Immediate data before change
Disassembled memory contents

To complete the command execution, input a slash and depress carriage return.

To display the contents of address 6

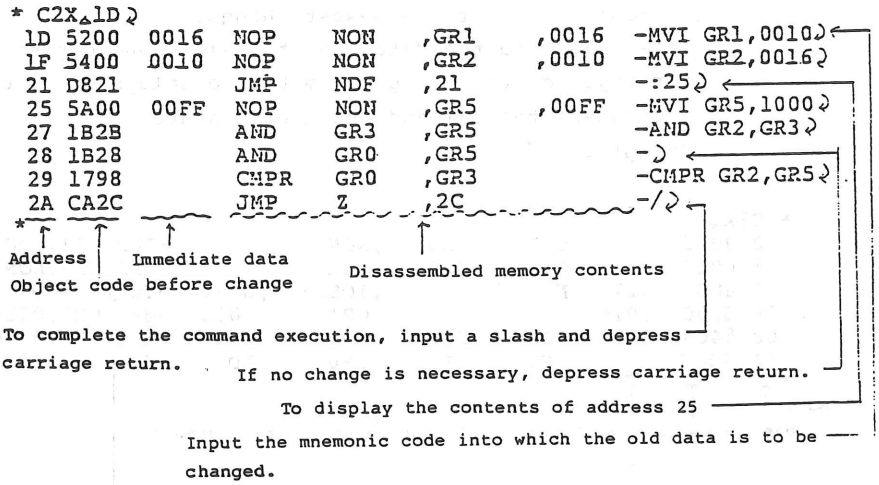
Input the mnemonic code into which the old data is to be changed.

b. When the prompt (*) is displayed, input the C2X command, a space, and a desired address followed by carriage return. The contents of IM2 from the specified address are then displayed as shown in the following program example. After the addresses, the object codes, and the disassembled contents of the memory have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the mnemonic code into which the old data is to be changed.

To change again the data already input, input the mnemonic code for the newest change.

After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in mnemonic format the contents of IM2 from address 1D



*For details on the mnemonic codes for IM2 (inputting format), refer to 4.4, IM2 assembler, in Chapter 4, Simple assembler syntax.

(5) CA command

```
* CAΔ [mm] [, z]↵
```

Changes in hexadecimal format the contents of RAM-A. The parameters of this command must satisfy this condition.
mm≤15, z≤F

a. When the prompt (*) is displayed, input the CA command followed by carriage return. The contents of RAM-A from address 0 of bank 00 are then displayed in hexadecimal format as shown in the program example below. After the banks, the addresses, and the data of RAM-A have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers.
To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

```
* CA↵
  00 00 00-F↵ ←—— Input the data into which the old data is to
  00 01 01-E↵      be changed.
  00 02 02-:E↵ ← To display the contents of address E
  00 0E 02-AD↵
  00 0F 0F-/↵ ← To complete the command execution, input a slash
*  ↑  ↑  ↑        and depress carriage return.
   |  |  Contents
   |  Address
   Bank
```

b. When the prompt (*) is diaplayed, input the CA command, a space, a comma, and a desired address followed by carriage return. The contents of RAM-A from the specified address of bank 00 are then displayed as shown in the following program example. After the banks, addresses, and data of RAM-A have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers.

To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change the contents of RAM-A from address D of bank 00

```
* CA␣,D↵
 00 0D 0D-1↵ ←—— Input data into which the old data is to be changed.
 00 0E 0E-↵  ←—— If no change is necessary, depress carriage return.
 00 0F 0F-:↵ ←—— To display the contents of the preceding address (to
 00 0E 0E-2↵     decrement the address by one)
 00 0F 0F-/↵ ←—— To complete the command execution, input a slash and
*↑  ↑  ↑         depress carriage return.
Bank Address Contents
```

c. When the prompt (*) is displayed, input the CA command, a space, a comma, and a desired bank followed by carriage return. The contents of RAM-A from address 00 of the specified bank are then displayed as shown in the following program example in hexadecimal format. After the banks, addresses, and data of RAM-A have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers.
After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.
Example: To change in hexadecimal format the contents of RAM-A from address 0 of bank 10

```
* CA 10↵
 10 00 00-FF↵ <-- Input the data into which the old data is to be changed.
 10 01 01-:F↵ <-- To display the contents of address F
 10 0F 0F-00↵
 10 00 FF-/↵  <-- To complete the command execution, input a slash and depress carriage return.
*  ↑  ↑  ↑
Bank  | Contents
   Address
```

d. When the prompt (*) is displayed, input the CA command, a space, a comma, a desired bank, and a desired address followed by carriage return. The contents of RAM-A from the specified address of the specified bank are then displayed in hexadecimal format as shown in the following program example. After the banks, addresses, and data of RAM-A have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers.

To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in hexadecimal format the contents of RAM-A from address A of bank 14

```
* CA␣14,A↵
 14 0A 4A-4F↵ <---- Input the data into which the old data is to
 14 0B 4B-4D↵       be changed.
 14 0C 4C-↵   <---- If no change is necessary, depress carriage
 14 0D 4D-30        return.
 14 0E 4E-/↵  <---- To complete the command execution, input a
*                   slash and depress carriage return.
 Bank | Contents
   Address
```

(6) CB command

```
* CBΔ [m] [, z]↵
```

Changes in hexadecimal format the contents of RAM-B. The parameters of this command must satisfy this condition.
m≤3, z≤F

a. When the prompt (*) is displayed, input the CB command followed by carriage return. The contents of RAM-B from address 0 of bank 0 are then displayed in hexadecimal format as shown in the example below. After the banks, addresses, and data of RAM-B have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers. To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.
Example:

```
* CB↵
 00 00 00-FF↵ ←—— Input the data into which the old data is
 00 01 01-FF↵      to be changed.
 00 02 02-/↵ ←—— To complete the command execution, input a
*                  slash and depress carriage return.
  ↑  ↑  ↑
  |Address|
Bank    Contents
```

b. When the prompt (*) is displayed, input the CB command, a space, a comma, and a desired address followed by carriage return. The contents of the RAM-B from the specified address of bank 0 are then displayed in hexadecimal format as shown in the following program example. After the banks, addresses, and data have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then, input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers.
To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.
Example: To change in hexadecimal format the contents of RAM-B from address A of bank 0

```
* CB␣,A↵
 00 0A 0A-1↵ ← Input the data into which the old data is to be changed.
 00 0B 0B-2↵
 00 0C 0C-:5↵ ← To display the contents of address 5
 00 05 05-F↵
 00 06 06-E↵
 00 07 07-/↵ ← To complete the command execution, input a slash
*                and depress carriage return.
Bank Address Contents
```

c. When the prompt (*) is displayed, input the CB command, a space, and a desired bank followed by carriage return. The contents of RAM-B from address 0 of the specified bank are then displayed in hexadecimal format as shown in the following program example. After the banks, addresses, and data of RAM-B have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state.

Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers.

To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in hexadecimal format the contents of RAM-B from address 0 of bank 2

```
* CB△2↵
 02 00 20-02↵ ←— Input data into which the old data is to be changed.
 02 01 21-↵   ←— If no change is necessary, depress carriage return.
 02 02 22-:↵  ←— To display the contents of the preceding address
 02 01 21-20↵     (i.e.,to decrement the address by one)
 02 02 22-/↵  ←— To complete the command execution,
*  ↑  ↑  ↑          input a slash and depress carriage return.
   Bank Address Contents
```

d. When the prompt (*) is displayed, input the CB command, a space, a desired bank, a comma, and a desired address followed by carriage return. The contents of RAM-B from the specified address of the specified bank are then displayed in hexadecimal format as shown in the following program example. After the banks, addresses, and data of RAM-B have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers.

To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in hexadecimal format the contents of RAM-B from address 2 of bank 3

```
* CB△3,2↵
  03 02 32-12↵ ←┐ Input the data into which the old data is to
  03 03 33-↵ ←──┘ be changed.
  03 04 34-ED↵    If no change is necessary, depress carriage
  03 05 35-:D↵←── return.
  03 0D 3D-33↵    To display the contents of
  03 0E 3E-/↵←──  address D.
  ↑  ↑  ↑         To complete the command execution, input a
  *  |  |         slash and depress carriage return.
Bank |  Contents
   Address
```

(7) CG command

```
* CG△[m] [,zz]↵
```

Changes in hexadecimal format the contents of RAM-G. The parameters of this command must satisfy this condition.

m<3, z<15

a. When the prompt (*) is displayed, input the CG command followed by carriage return. The contents of RAM-G from address 00 of bank 0 are then displayed in hexadecimal format as shown in the following program example. After the banks, addresses, and data of RAM-G have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 4 digits of hexadecimal numbers. To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example:

```
* CG↵
 00 00 0000-AA00↵
 00 01 0001-↵
 00 02 0002-BB00↵
 00 03 0003-:10↵
 00 10 0010-1000↵
 00 11 0011-↵
 00 12 0012-2000↵
 00 13 0013-/↵
*
```

Input the data into which the old data is to be changed.

If no change is necessary, depress carriage return.

To display the contents of address 10.

To complete the command execution, input a slash and depress carriage return.

Bank | Contents
Address

b. When the prompt (*) is displayed, input the CG command, a space, a comma, and a desired address followed by carriage return. The contents of RAM-G from the specified address of bank 0 are then displayed in hexadecimal format as shown in the following program example. After the banks, addresses, and data of RAM-G have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 4 digits of hexadecimal numbers.
To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.
Example: To change in hexadecimal format the contents of RAM-G from address 13 of bank 0

```
* CG␣,13↵
  00 13 0013-0000↵ ←Input the data into which the old data is
  00 14 0014-0000↵  to be changed.
  00 15 0015-/↵ ←To complete the command execution, input
* ↑  ↑  ↑        a slash and depress carriage return.
Bank Address Contents
```

c. When the prompt (*) is displayed, input the CG command, a space, and a desired bank followed by carriage return. The contents of RAM-G from address 00 of the specified bank are then displayed in hexadecimal format as shown in the following program example. After the banks, addresses, and data of RAM-G have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 4 digits of hexadecimal numbers.

To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

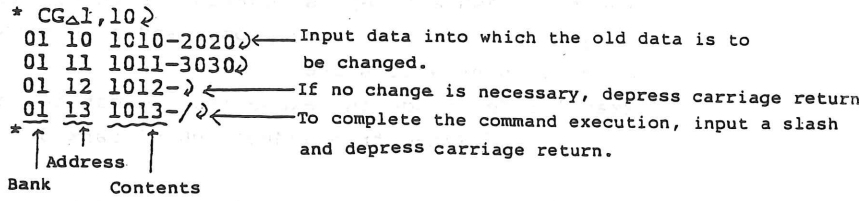
Example: To change in hexadecimal format the contents of RAM-G from address 00 of bank 3

```
* CG△3↵
 03 00 3000-F000↵←┐ Input the data into which the old data is to
 03 01 3001-↵ ←──┘ be changed.
 03 02 3002-E045↵  If no change is necessary, depress carriage
 03 03 3003-:E↵←── return.
 03 0E 300E-0000↵  To display the contents
 03 0F 300F-/↵ ←── of address E
                   To complete the command execution, input a
                   slash and depress carriage return.
 *  ↑  ↑    ↑
Bank Address Contents
```

d. When the prompt (*) is displayed, input the CG command, a space, a desired bank, a comma, and a desired address followed by carriage return. The contents of RAM-G from the specified address of the specified bank are then displayed in hexadecimal format as shown in the following program example. After the banks, addresses, and data of RAM-G have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 4 digits of hexadecimal numbers.

To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in hexadecimal format the contents of RAM-G from address 10 of bank 1



(8) CR command

* CR△[RNAME]↵

Changes in hexadecimal format the contents of the internal registers. However, the contents of the PC1, GR7, MC, and PC2F registers cannot be changed by this command.

a. When the prompt (*) is displayed, input the CR command followed by carriage return. The contents of the internal registers are then displayed in hexadecimal format, as shown in the following program example, starting from the IR register.
After the register names and their data have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the register contents are to be changed as a maximum of 4 digits of hexadecimal numbers.

To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example:

```
* CR ↵
IR      0001-F001↵ ←——Input the data into which the old data is
GR0     0003-0001↵      to be changed.
GR1     0004-0002↵
GR2     0005-0003↵
GR3     0006-9991↵
GR4     0007-:↵ ←————To display the contents of the preceding
GR3     9991-0004↵      register
GR4     0007-:WCNTA↵ ←——To display the contents of the WCNTA register
WCNTA   0006-1000↵
WCNTB   0007-2000↵
IR      F001-/↵ ←———To complete the command execution, input a slash
*                      and depress carriage return.
```

Register name Contents

b. When the prompt (*) is displayed, input the CR command, a space, and a desired register name followed by carriage return. The contents of the internal registers are then displayed in hexadecimal format, starting from the specified register. After the register names and their data have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the register contents are to be changed as a maximum of 4 digits of hexadecimal numbers.
To change again the data already input, input the data for the newest change as a maximum of 4 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in hexadecimal format the contents of the internal registers, starting from the DRG0 register

```
* CR△DRG0↵
DRG0   000D-010F↵ ←— Input the data into which the old data is to
DRG1   C00E-01AC↵      be changed.
IOSR1  000F-↵ ←———— If no change is necessary, depress
ICSR2  0010-↵          carriage return.
PCNTA  C011-0000↵
PCNTB  0012-↵
BAR    0102-/↵ ←———— To complete the command execution, input a
*                     slash and depress carriage return.
```

Register name

Contents

*The contents of the internal registers are changed from the IR register to the WCNTB register in the order in which the register names are listed in Table 1.3. However, the program skips and does not display the registers whose contents cannot be changed (i.e., registers PC1, GR7, MC, and PC2F).

(9) CP command

```
* CP△[xxxx]↵
```

Changes in hexadecimal format the contents of the pattern memory (PMEM). The parameters of this command must satisfy the condition.

xxxx≤FFFF

a. When the prompt (*) is displayed, input the CP command followed by carriage return. The contents of PMEM from address 00 are then displayed in hexadecimal format as shown in the following program example. After the addresses and data of PMEM have been displayed, a hyphen is displayed, indicating that the program is in the command input wait state. Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers.
To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.
Example:

```
* CP ↵
0000 34-46 ↵ ←— Input the data into which the old data is to
0001 AD-FF ↵      be changed.
0002 00- ↵ ←— If no change is necessary, depress carriage return.
0003 A1-01 ↵
0004 00-:A000 ↵ ←— To display the contents of address A000
A000 FF-: ↵ ←— To display the contents of the preceding address
9FFF 00-12 ↵
A000 FF-/ ↵ ←— To complete the command execution, input a slash
*                  and depress carriage return.
  ↑    ↑
Address Contents
```

b. When the prompt (*) is displayed, input the CP command, a space, and a desired address followed by carriage return. The contents of PMEM from the specified address are then displayed in hexadecimal format as shown in the following program example. Then input the data into which the old data is to be changed as a maximum of 2 digits of hexadecimal numbers. To change again the data already input, input the data for the newest change as a maximum of 2 digits of hexadecimal numbers. After changing the data, input a slash and depress carriage return. The program then displays the prompt and enters the command input wait state.

Example: To change in hexadecimal format the contents of PMEM from address 4800

```
* CP␣4800↵
 4800 23-8A↵ ← Input the data into which the old data is to be changed.
 4801 2A-↵ ←— If no change is necessary, depress carriage return.
 4802 AA-3B↵
 4803 00-:FF23↵ ←— To display the contents of address FF23
 FF23 00-12↵
 FF24 00-35↵
 FF25 00-/↵ ←— To complete the command execution, input a slash
*                  and depress carriage return.
```

Address Contents

3.2.3 GO command

```
* GOΔ [xx] [, yy] [, LLLL]↵
```

Executes the program for the μPD7764 currently stored in the memory.

(1) When no break point is specified

```
* GOΔ [xx]↵
```

The parameter of this command must satisfy this condition.

xx≤7F (except for 7D)

When the prompt (*) is displayed, input the GO command, a space, and a desired address followed by carriage return. The user program is then executed from the specified address.

If specifying the address is omitted by the GO command being input after the prompt and followed immediately by carriage return, the user program execution will start from address 00.

*When this command is executed, address 7D of instruction memory 2 (IM2) is used by the system and therefore cannot be used by the user program.

(2) When a break point is specified

```
* GOΔ [xx], yy [, LLLL]↵
```

The parameters of this command must satisfy these conditions.

xx≤7F (except for 70 to 7D), yy≤6F

Because parameter yy is treated as the break point, it must not be omitted.

When the prompt (*) is displayed, input the GO command, a space, a desired start address, a comma, an address that is used as the break point, a comma and then the number of times the program is to be executed. Conclude the input with carriage return. The user program is then executed from the specified address until it passes the specified break point the specified number of times. Unless the start address is specified, address 00 is assumed. If specifying the break point is omitted, 1 is assumed and thus the program is executed only once.

*When this command is ececuted, addresses 70 to 7D of IM2 are used by the system and therefore cannot be used by the user program.

When setting a break point, the set break point must satisfy these conditions.

(a) The address at which the break point is set must be within addresses 00 through 6F of the RAM area.

(b) The break point cannot be set at an address where the JUMP or RETURN instruction is stored.

(c) A break occurs after the instruction at the address where the break point is set has been executed. For this reason, if the break point is set at an address where the CALL instruction is stored, the break occurs after the subroutine specified by the CALL instruction has been executed.

(d) If the program execution passes the break point when the contents of the RQM flag are other than 00, the following operations are performed.

   a. If the program execution passes the break point when the contents of the RQM flag are 11, the message MODE ERROR is displayed on the console and the program execution is aborted. The program then displays the prompt and enters command input wait state.

b. If the program execution passes the break point when the contents of the RQM flag are 01 or 10, the RQM flag is cleared to 00 because the μPD7764 uses the flag when the program execution passes the break point.

(3) Operation after inputting GO command

(a) Setting of parameters

When the GO command is executed, the following message is displayed and the program is waiting for parameters to be input.

SET PARAMETER OR RETURN
?

Under this condition, the following three parameters can be input.

/D : This serves as an ON-OFF switch for dumping the internal register contents when the program execution passes the break point. When the program execution passes the break point with this switch set to ON, the contents of the internal registers immediately after the program execution has passed the break point are dumped and listed in the same manner as when the DR command is executed. (Because the default condition of this parameter is OFF, to turn it ON, specify /D only once. If the parameter is specified twice, the ON state of the switch is changed into the OFF state.)

/R : This parameter causes the current contents of the SR register to be displayed in bit pattern.

Display example:

```
SR = 00000001
     ↑
    MSB
```

/ : This parameter causes the execution of the GO command to be aborted and the prompt to be displayed; thus, the program enters the command input wait state.

Plural number of parameters can be simultaneously set. If a specific parameter is not specified, only depress carriage return.

(b) Controlling D-PART

When the break point is set, the following message is displayed on the console and the program enters the command input wait state.

D-PROCESSOR RESTARTS AFTER BREAK (Y OR OTHER)
?

If the D-PART (IM1) is in operation when the program execution passes the break point, input Y followed by carriage return. If not, only depress carriage return.

*If any character other than Y is input when the program execution passes the break point with the D-PART (IM1) in operation, the D-PART will not operate after the program execution has passed the break point.

(c) Service at program execution

After the settings described in (a) and (b) above have been performed, the following message is displayed and execution of the μPD7764 program starts. Then the functions described below can be utilized.

SRP STARTS

a. Data I/O function via DR register

While executing the program, the μPD7764 displays nothing on the console. However, the following message is displayed on the console if I/O of data is requested via the DR register when the state of the RQM flag in the SR register has changed.

. When data output is requested

1-byte data

```
SR = 00001001 NO.      1 SRP BYTE OUT # ED
```

MSB — SR register bits — No. of times 1-byte data is output (decimal) — Output data (hexadecimal)

1-word data

```
SR = 00000001 NO.      1 SRP WORD OUT # 0000
```

MSB — SR register bits — No. of times 1-word data is output (decimal) — Output data (hexadecimal)

. When data input is requested

1-byte data

```
SR = 00001010 NO.      1 SRP BYTE IN  ? 12↵
```

MSB — SR register bits — No. of times 1-byte data is output (decimal) — The program waits for data input. Input data as a 4-digit hexadecimal number.

1-word data

```
SR = 00000010 NO.      1 SRP WORD IN  ? FD34↵
```

MSB — SR register bits — No. of times 1-word data is output (decimal) — The program waits for data input. Input data as a 4-digit hexadecimal number.

The parameters described in paragraph (a) can be input after the data.

```
SR = 00001010 NO.      2 SRP BYTE IN  ? 34/D↵
```

The parameters can be input after data.

If data is erroneously input, this message is displayed requesting for reinput of data. Input the correct data. In this case, the parameters cannot be input.

```
DATA ERROR
?
```

If a word exceeding the valid number of digits is input, the latter 4 digits of the word are valid. If a byte exceeding the valid number of digits is input, the latter 2 digits are valid. If data short of the valid number of digits is input, the higher digit(s) is regarded as 0.

b. Changing parameters

The parameters described in paragraph (a) can be changed during program execution by inputting a new parameter from the console. For example, when parameter /D is in the ON state, inputting this parameter again from the console causes the ON state to turn to the OFF state.

(4) Dumping register contents immediately after program execution has passed break point

When the program execution passes the break point with parameter /D set to ON, the contents of the internal registers are dumped and listed, as when the DR command is executed, immediately after the program execution has passed the break point. However, the contents of the IR register cannot be correctly read and therefore, FFFF is displayed instead. Also, the contents of the PC2F register of the program counter (PC2) are not correctly displayed.

If the D-PART (IM1) is in operation when the program execution passes the break point, the D-PART (IM1) break is delayed. For this reason, the contents of the internal registers immediately after the program execution has passed the break point are not displayed.

(5) How to return program from executing GO command to command input wait state

The following three methods can be used to return the program from executing the GO command to the command input wait state.

a. When setting the parameter, input a slash and depress carriage return. Then execution of the program is not started and thus the GO command is aborted. The program then displays the prompt and enters the command input wait state.

b. During execution of the program (when changing the parameter), input a slash and depress carriage return. Then current execution of the program is aborted and the prompt is displayed indicating that the program enters the command input wait state. If the program execution is forcibly aborted by inputting a slash and depressing carriage return, the IM1 contents when the program execution has started can be referenced. The contents of IM2, all the other RAMs, and internal registers which can be referenced are the same contents as those immediately after the slash followed by carriage return. The correct contents of internal registers PC1, GR7, MC, and PC2F cannot be read because the program execution has been forcibly aborted. Therefore, FFFF is displayed on the console in place of their contents.

c. If a break point has been set when inputting the GO command, the following message is displayed and the program enters the command input wait state when the program execution passes the break point the specified number of times (i.e., when the break condition is satisfied).

Under this condition, inputting 1 or 2 each followed by carriage return causes the prompt to be displayed indicating that the program is in the command input wait state. If 1 is input followed by carriage return, the contents of IM1 will be those when IM2 satisfies the break condition, and the contents of IM2 will be those when the program execution has started.

If 2 is input followed by carriage return, the contents of IM2 will be those when the break condition is satisfied, and the contents of IM1 will be those when the program execution has started.

In either case, the contents of all the RAMs and internal registers will be those when the break condition is satisfied (i.e., the contents immediately after the instruction at the break point has been executed). However, the contents of internal register PC2F of the program counter (PC2) will not be correctly displayed.

If the D-PART (IM1) is in operation when the break condition is satisfied, the D-PART (IM1) break is delayed. For this reason, the contents of the internal registers immediately after the program execution has passed the break point are not displayed.

*The contents when the program execution started are the contents at initialization (i.e., when the program is loaded to the μPD7764 by resetting it).

(6) Execution example of GO command

Example 1: To execute the program from address 00 without setting the break point

```
* GO ↵
SET PARAMETER OR RETURN
? ↵ ←———————— No parameter is set.

SRP STARTS ———————— The GO command is aborted.
/↵
* DR↵

< REGISTER DUMP >
IR   - 0000  PC1  - FFFF  GR0  - 0000  GR1  - 0000  GR2  - 0000
GR3  - 005F  GR4  - 0000  GR5  - 00FF  GR6  - 0000  GR7  - FFFF
GARG - 0000  MC   - FFFF  DRG0 - 0000  DRG1 - 0000  IOSR1 - 1070
IOSR2 - 0000 PCNTA - 0001 PCNTB - 0000 PC2F - FFFF  BAR  - 160A
LCNT - 0000  RCNT - 0000  WCNTA - 0000 WCNTB - 0000
*
```

When execution of the GO command is aborted and the register contents are dumped by executing the DR command, FFFF is displayed in place of the contents of the PC1, GR7, MC, and PC2F registers.

Example 2: To execute the program from address 00 without setting the break point

```
* GO ↵
SET PARAMETER OR RETURN
? ↵ ←———————— No parameter is set.

SRP STARTS
SR = 00001010 NO.    1 SRP BYTE IN  ? 2A ↵ }
SR = 00001010 NO.    2 SRP BYTE IN  ? 4F ↵ } Data input
SR = 00000010 NO.    1 SRP WORD IN  ? FF3A ↵ }
SR = 00000001 NO.    1 SRP WORD OUT = 0000
SR = 00000001 NO.    2 SRP WORD OUT = 0000
/↵ ←——— The GO command is aborted.
*
```

If the state of the RQM flag is changed during program execution, input of commands is requested as shown above.

Example 3: To execute the program from address 00 until it passes address 04 once where the break point is set

```
* GO△0,4 ↵
SET PARAMETER OR RETURN
? /↵
*
```

The GO command is aborted without starting program execution.

Example 4: To execute the program from address 00 until it passes address 0D twice where the break point is set

```
* GO△0,D,2 ↵
SET PARAMETER OR RETURN
? /D ↵ ←———————— Parameter /D is specified

D-PROCESSOR RESTARTS AFTER BREAK (Y OR OTHER) ← Input Y followed by carriage
? Y↵ ←                                          return because the D-PART is
                                                in operation when the program
SRP STARTS                                      execution passes the break point.

< REGISTER DUMP >
IR    - FFFF  PC1   - 0023  GR0   - 0001  GR1   - 0016  GR2   - 000F
GR3   - 005F  GR4   - 0000  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0170
IOSR2 - 0000  PCNTA - 0001  PCNTB - 0000  PC2F  - 00D3  BAR   - 160A
LCNT  - 0000  RCNT  - 0000  WCNTA - 0001  WCNTB - 0000

< REGISTER DUMP >
IR    - FFFF  PC1   - 0023  GR0   - 0002  GR1   - 0016  GR2   - 000E
GR3   - 005F  GR4   - 0000  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0170
IOSR2 - 0000  PCNTA - 0001  PCNTB - 0000  PC2F  - 00D3  BAR   - 160A
LCNT  - 0000  RCNT  - 0000  WCNTA - 0002  WCNTB - 0000
IM-1 dump or IM-2 dump (1/2) ? 2↵
*
```

Example 5: To execute the program from address 00 until it passes address 11 twice where the break point is set

```
* GO△0,11,2 ↵
SET PARAMETER OR RETURN
? ↵ ←———————— No parameter is set.

D-PROCESSOR RESTARTS AFTER BREAK (Y OR OTHER)
? Y↵ ←———————— Input Y followed by carriage
                return because the D-PART is
SRP STARTS      in operation when the program
IM-1 dump or IM-2 dump (1/2) ? 2↵  execution passes the break point.
```

The result of the program execution will be as shown above unless parameters are set.

Example 6: To execute the program from address 00 until it passes address 40 five times where the break point is set

```
* GO△0,40,5 ↵
SET PARAMETER OR RETURN
? /D↵ ←———————— Parameter /D is specified

D-PROCESSOR RESTARTS AFTER BREAK (Y OR OTHER)
? N↵ ←———————— Input other than Y because
                the D-PART is not in operation
SRP STARTS      when the program execution
                passes the break point.
< REGISTER DUMP >
IR    - FFFF  PC1   - 0020  GR0   - 8000  GR1   - 0000  GR2   - 0000
GR3   - 0000  GR4   - 0000  GR5   - 0000  GR6   - 0000  GR7   - 9876
GARG  - 00C0  MC    - 0001  DRG0  - 0002  DRG1  - 0003  IOSR1 - 0004
IOSR2 - 0005  PCNTA - 0006  PCNTB - 0007  PC2F  - 1AD3  BAR   - 0000
LCNT  - 0000  RCNT  - 0000  WCNTA - 0000  WCNTB - 0000

< REGISTER DUMP >
IR    - FFFF  PC1   - 0020  GR0   - 903E  GR1   - 0000  GR2   - 0000
GR3   - 0000  GR4   - 0000  GR5   - 0000  GR6   - 0000  GR7   - 9876
GARG  - 00C0  MC    - 0001  DRG0  - 0002  DRG1  - 0003  IOSR1 - 0004
IOSR2 - 0005  PCNTA - 0006  PCNTB - 0007  PC2F  - 10D3  BAR   - 0000
LCNT  - 0000  RCNT  - 0000  WCNTA - 0000  WCNTB - 0000
/↵ ←——— The GO command is aborted.
* DR↵ ←——— The DR command is executed.

< REGISTER DUMP >
IR    - 0000  PC1   - FFFF  GR0   - 903E  GR1   - 0000  GR2   - 0000
GR3   - 0000  GR4   - 0000  GR5   - 0000  GR6   - 0000  GR7   - FFFF
GARG  - 00C0  MC    - FFFF  DRG0  - 0002  DRG1  - 0003  IOSR1 - 0004
IOSR2 - 0005  PCNTA - 0006  PCNTB - 0007  PC2F  - FFFF  BAR   - 0000
LCNT  - 0000  RCNT  - 0000  WCNTA - 0000  WCNTB - 0000
*
```

In the same way as Example 1, FFFF is displayed in place of the contents of the PC1, GR7, MC, and PC2F registers.

Example 7: To execute the program from address 00 until it passes address 2D three times where the break point is set.

```
* GO,0,2D,3↵
SET PARAMETER OR RETURN
? /D↵ ←———————— Parameter /D is specified.

D-PROCESSOR RESTARTS AFTER BREAK (Y OR OTHER)
? Y↵←——— Input Y followed by carriage return because the D-PART is in
SRP STARTS operation when the program execution passes the break point.

< REGISTER DUMP >
IR    - FFFF  PC1   - 002D  GR0   - 0001  GR1   - 0016  GR2   - 000F
GR3   - 0000  GR4   - 0000  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0000
IOSR2 - 01D0  PCNTA - 0001  PCNTB - 0000  PC2F  - 00D3  BAR   - 090A
LCNT  - 160D  RCNT  - 0003  WCNTA - 0000  WCNTB - 0000

< REGISTER DUMP >
IR    - FFFF  PC1   - 002D  GR0   - 0002  GR1   - 0016  GR2   - 000E
GR3   - 0001  GR4   - 0000  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0000
IOSR2 - 01D0  PCNTA - 0001  PCNTB - 0000  PC2F  - 00D3  BAR   - 090A
LCNT  - 160C  RCNT  - 0004  WCNTA - 0000  WCNTB - 0000
/R↵ ←———————— Parameter /R is input to change the existing parameter.
SR = 00000011

< REGISTER DUMP >
IR    - FFFF  PC1   - 002D  GR0   - 0003  GR1   - 0016  GR2   - 000D
GR3   - 0002  GR4   - 0000  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0000
IOSR2 - 01D0  PCNTA - 0001  PCNTB - 0000  PC2F  - 00D3  BAR   - 090A
LCNT  - 160B  RCNT  - 0005  WCNTA - 0000  WCNTB - 0000
IM-1 dump or IM-2 dump (1/2) ? 2
*
```

3.2.4 USER command

```
* USR↵
```

Connects the μPD7764 bus on the EVAKIT-7764 to the user system. In the user mode, the μPD7764 and its pattern memory on the EVAKIT-7764 can be freely used from the user system side.
Also, since the EVAKIT-7764 is provided with an FIFO (First-In, First-Out) memory that is activated by the reset signal of the μPD7764, the initial program of the μPD7764 can always be monitored.
When the prompt (*) is displayed, input the USR command followed by carriage return. The program displays the following message and enters the command input wait state.

```
*** USER MODE ***
"PRESS" '/'TO STOP USER MODE
```

The μPD7764 and its pattern memory can then be freely used from the user system side until a slash is input followed by carriage return.
If a slash is input followed by carriage return, the EVAKIT-7764 resets the μPD7764 and displays the following message. The bus is then connected to the EVAKIT-7764 and the program enters the command input wait state.

```
IM-2 DUMP-BEFORE(1) / AFTER(2)EXECUTION (1/2) ?
```

When this message is displayed, either 1 or 2 can be input. When 1 is input followed by carriage return, the contents of IM2 will be those which has been input to the μPD7764 from the user system. If 2 is input followed by carriage return, the contents of IM2 will be those immediately after the slash has been input followed by carriage return.

In either case, the contents of IM1 will be those which have been input to the μPD7764 from the user system.
The contents of all RAMs and internal registers will always be those when the slash has been input followed by carriage return. The contents of internal registers PC1, GR7, MC, and PC2F will be lost immediately after the μPD7764 has accepted the input slash and carriage return. Therefore, FFFF is displayed in place of the contents of these registers.
Execution example of user command:

```
* USR↵
*** USER MODE ***
"PRESS" '/'TO STOP USER MODE.
/↵ ←
IM-2 DUMP-BEFORE(1) / AFTER(2)EXECUTION (1/2) ? 1↵
*
```

Input 1 followed by carriage return to transfer to the EVAKIT the memory contents that have been input from the user system to the μPD7764.

Input a slash followed by carriage return to change the mode from the user mode to EVAKIT mode.

NOTE:
The program developed by the EVAKIT-7764 cannot be transferred to the user system.
REFERENCE:
The best way to use this command is considered as follows:

1. Input the USR command to connect the μPD7764 bus to the user system.
2. Load the program from the user system to the μPD7764.
3. Input a slash followed by carriage return to connect the μPD7764 bus to the EVAKIT-7764.
4. The program loaded to the μPD7764 from the user system now can be referenced, modified, or executed at the EVAKIT-7764 side.

3.2.5 SAVE command

Outputs the contents of the μPD7764 internal registers and RAMs to the RS-232C interface

(1) S command

```
* S△A↵
```

Outputs the contents of IM1 and IM2

Command input example

```
* S△A↵
```

(2) SW command

```
* SW△A↵
```

Outputs the contents of the internal registers, RAM-A, RAM-B, and RAM-G

Command input example

```
* SW△A↵
```

(3) SP command

```
* SP△A;xxxx,yyyy↵
```

Outputs the contents of PMEM

Command input example (To output the PMEM contents from address 100 to FFFF)

```
* SP△A,100,FFFF↵
```

To use the SAVE command effectively, a system that can be controlled in accordance with the transfer protocol as shown below is required at the host system side to establish data communication.

For details on the file format used to output data, refer to Appendix 3, Transfer protocol.

Transfer protocol

When the EVAKIT-7764 receives the SAVE command, it waits for 02H code, requesting for data transfer, to be sent from the host system.

When the EVAKIT-7764 receives the 02CH code, it outputs physical records in units of 128 bytes to the host system, and waits for 17H code that will be transferred from the host system and serve as an acknowledge signal for data reception. When the EVAKIT-7764 receives the 17H code, it waits for 02H code requesting for the next data to be sent or 1AH code indicating completion of the data transfer. Upon reception of the 1AH code, the EVAKIT-7764 completes the execution of the SAVE command. The program then displays the prompt and enters the command input wait state.

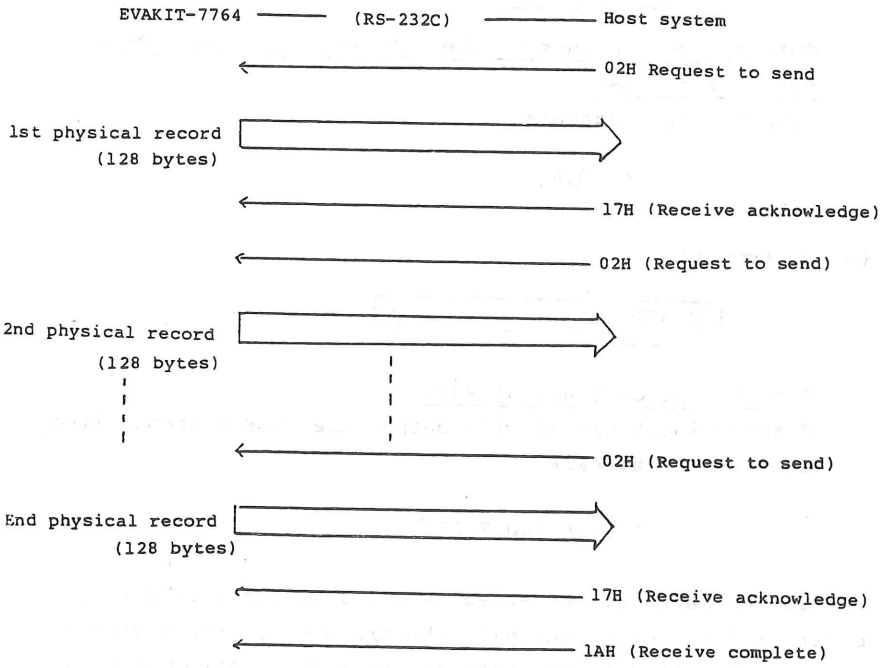


Fig. 3-1 SAVE Command Transfer Protocol

3.2.6 LOAD command

Inputs data from the RS-232C interface to the μPD7764 internal registers and RAMs

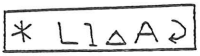
(1) L command

Inputs data to IM1 and IM2

Command input example

```
* L△A↵
```

(2) L1 command

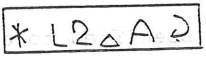


Inputs data to IM1

Command input example

```
* L1△A↵
```

(3) L2 command



Inputs data to IM2

Command input example

```
* L2△A↵
```

(4) LW command

```
* LW△A↵
```

Inputs data to the internal registers, RAM-A, RAM-B, and RAM-G

Command input example

```
* LW△A↵
```

(5) LP command

```
* LP△A[,XXXX]↵
```

Inputs data to PMEM

xxxx : offset load address

When the offset load address is specified, data are input to PMEM from the specified offset load address. If an attempt is made to load data exceeding address FFFF, the exceeding data will be ignored.

Command input example:

a) Without specifying the offset load address

```
* LP△A↵
```

b) With offset load address specified to address 2000

```
* LP△A,2000↵
```

To use the LOAD command effectively, a system that can be controlled in accordance with the transfer protocol as shown below is required at the host system side to establish data communication. For details on the file format used to input data, refer to Appendix 3, Transfer protocol.

Transfer protocol

When the EVAKIT-7764 receives the LOAD command, it outputs 02H code as a transfer request signal and waits for physical records to be transferred from the host system. The physical records are sent in units of 128 bytes.

Upon reception of the 128-byte data (i.e., physical records) from the host system, the EVAKIT-7764 outputs the 17H code serving as an acknowledge signal for data reception and then outputs to the host system the 02H code requesting for the next data to be transferred, or the 1AH code indicating the completion of data reception. After transferring the 1AH code to the host system, the EVAKIT-7764 completes the execution of the LOAD command. The program then displays the prompt and enters the command input wait state.

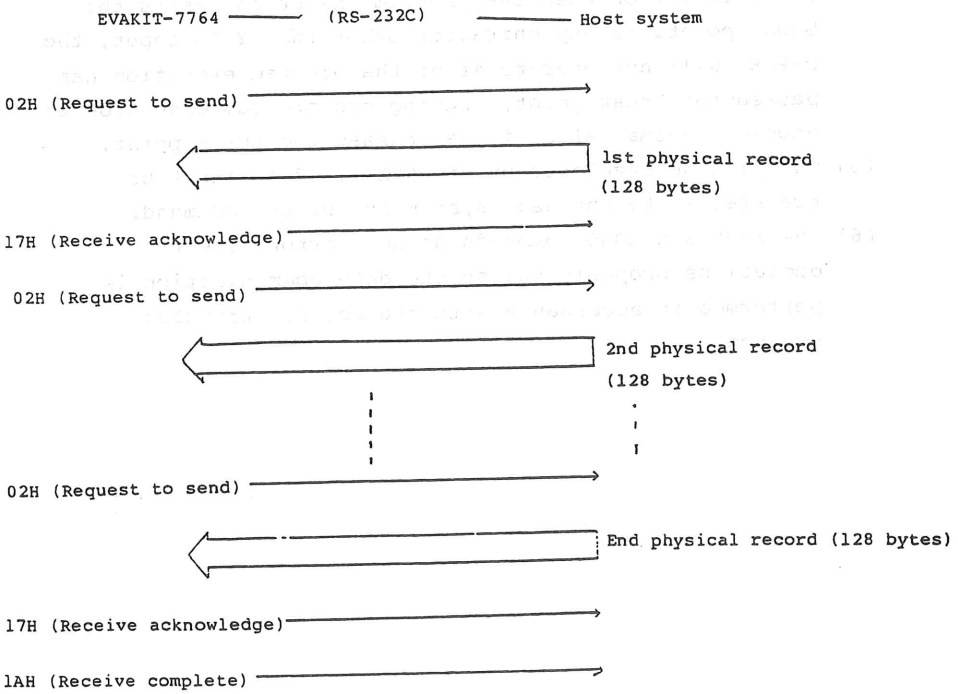


Fig. 3-2 LOAD Command Transfer Protocol

3.3 Notes on Using Commands

(1) The contents of the PC1, GR7, MC, and PC2F registers cannot be changed by executing the CR command.

(2) When the GO command is executed, the PC1, MC, PC2F, DR, IOB, and SR registers are reset because the μPD7764 is reset and the program is loaded to it.

(3) Occurrence of break is delayed when the D-PART (IM1) is in operation.

(4) If the break point is set during control of the D-PART, this message is displayed and the program enters the command input wait state.

```
D-PROCESSOR RESTARTS AFTER BREAK (Y OR OTHER)
?
```

Input Y followed by carriage return if the D-PART (IM1) is in operation when the program execution passes the break point. If any character other than Y is input, the D-PART will not operate after the program execution has passed the break point, causing the program execution to become abnormal after it has passed the break point.

(5) The program developed by the EVAKIT-7764 cannot be transferred to the user system by the USR command.

(6) The SAVE and LOAD commands do not perform their operations properly unless the data communication is performed in accordance with the correct transfer protocol.

3.4 Command Execution Examples

This section shows examples of executing the monitor commands.

```
* USR↵ <------------------ The USR command is used to load the program on
*** USER MODE ***            the user system to the EVAKIT.
"PRESS" '/'TO STOP USER MODE
/↵
IM-2 DUMP-BEFORE(1) / AFTER(2)EXECUTION (1/2) ? 1↵
* D1↵ <---- To check the contents of IM1

< DUMP IM1 >
00 :  E000  2800  7800  0040  9001  8000  D009  C00A
08 :  B016  A00F  0140  2800  7800  1000  0180  2800
10 :  7800  1800  2800  7800  D016  B016  6800  7000
18 :  1840  2800  7800  0000  0000  0000  0000  0000
* D2↵ <---- To check the contents of IM2

< DUMP IM2 >
00 :  001C  0018  6800  0170  915E  D805  5200  0016
08 :  5400  0010  904C  D60B  0028  003A  C80A  800E
10 :  D810  0039  C808  0018  914C  6A00  01D0  4A00
18 :  0005  D819  8009  800E  0018  5200  0016  5400
20 :  0010  D821  88BC  D623  800E  5A00  00FF  1B2B
28 :  1B28  1798  CA2C  002C  0028  003A  C821  0039
30 :  C81F  0018  D832  915C  9040  D635  6800  1070
38 :  800E  D839  905C  9820  DE3C  C03D  0000  0000
40 :  0000  0000  0000  0000  0000  0000  0000  0000
48 :  0000  0000  0000  0000  0000  0000  0000  0000
50 :  0000  0000  0000  0000  0000  0000  0000  0000
58 :  0000  0000  0000  0000  0000  0000  0000  0000
60 :  0000  0000  0000  0000  0000  0000  0000  0000
68 :  0000  0000  0000  0000  0000  0000  0000  0000
70 :  0000  0000  0000  0000  0000  0000  0000  0000
78 :  0000  0000  0000  0000  0000  0000  0000  0000
* DR↵ <---- To check the contents of the registers

< REGISTER DUMP >
IR    - 0000  PC1   - 0000  GR0   - 0000  GR1   - 0000  GR2   - 0000
GR3   - 0000  GR4   - 0000  GR5   - 0000  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0000
IOSR2 - 0000  PCNTA - 0000  PCNTB - 0000  PC2F  - 0000  BAR   - 0000
LCNT  - 0000  RCNT  - 0000  WCNTA - 0000  WCNTB - 0000
* GO↵ <------------------ To execute the program
SET PARAMETER OR RETURN
?↵

SRP STARTS
SR = 00000001 NO.      1 SRP WORD OUT = 0000
/↵
* D1↵ <---- To check the contents of IM1

< DUMP IM1 >
00 :  E000  2800  7800  0040  9001  8000  D009  C00A
08 :  B016  A00F  0140  2800  7800  1000  0180  2800
10 :  7800  1800  2800  7800  D016  B016  6800  7000
18 :  1840  2800  7800  0000  0000  0000  0000  0000
* DR↵ <---- To check the contents of the registers

< REGISTER DUMP >
IR    - 0000  PC1   - FFFF  GR0   - 0000  GR1   - 0000  GR2   - 0000
GR3   - 005F  GR4   - 0000  GR5   - 00FF  GR6   - 0000  GR7   - FFFF
GARG  - 0000  MC    - FFFF  DRG0  - 0000  DRG1  - 0000  IOSR1 - 1070
IOSR2 - 0000  PCNTA - 0000  PCNTB - 0000  PC2F  - FFFF  BAR   - 160A
LCNT  - 0000  RCNT  - 0000  WCNTA - 0000  WCNTB - 0000
*
```

```
* D2 ↵ ←——— To check the contents of IM2

< DUMP IM2 >
00 : 001C 0018 6800 0170 915E D805 5200 0016
08 : 5400 0010 904C D60B 0028 003A C80A 800E
10 : D810 0039 C808 0018 914C 6A00 01D0 4A00
18 : 0005 D819 8009 800E 0018 5200 0016 5400
20 : 0010 D821 88BC D623 800E 5A00 00FF 1B2B
28 : 1B28 1798 CA2C 002C 0028 003A C821 0039
30 : C81F 0018 D832 915C 9040 D635 6800 1070
38 : B0DE D839 905C 9820 DE3C C03D 0000 0000
40 : 0000 0000 0000 0000 0000 0000 0000 0000
48 : 0000 0000 0000 0000 0000 0000 0000 0000
50 : 0000 0000 0000 0000 0000 0000 0000 0000
58 : 0000 0000 0000 0000 0000 0000 0000 0000
60 : 0000 0000 0000 0000 0000 0000 0000 0000
68 : 0000 0000 0000 0000 0000 0000 0000 0000
70 : 0000 0000 0000 0000 0000 0000 0000 0000
78 : 0000 0000 0000 0000 0000 C000 0000 0000
* GO 0,2F,2 ↵ ←——— To execute the program
SET PARAMETER OR RETURN
? /D ↵

D-PROCESSOR RESTARTS AFTER BREAK (Y OR OTHER)
? Y ↵

SRP STARTS

< REGISTER DUMP >
IR    - FFFF  PC1   - 002D  GR0   - 0010  GR1   - 0015  GR2   - 0000
GR3   - 000F  GR4   - 0000  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0000
IOSR2 - 01D0  PCNTA - 0000  PCNTB - 0000  PC2F  - 00D3  BAR   - 090A
LCNT  - 150E  RCNT  - 0012  WCNTA - 0000  WCNTB - 0020

< REGISTER DUMP >
IR    - FFFF  PC1   - 002D  GR0   - 0020  GR1   - 0014  GR2   - 0000
GR3   - 001F  GR4   - 0000  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0000
IOSR2 - 01D0  PCNTA - 0000  PCNTB - 0000  PC2F  - 00D3  BAR   - 090A
LCNT  - 140E  RCNT  - 0022  WCNTA - 0000  WCNTB - 0020
IM-1 dump or IM-2 dump (1/2) ? 2 ↵
* D1 ↵ ←——— To check the contents of IM1

< DUMP IM1 >
00 : E000 2800 7800 0040 9001 8000 D009 C00A
08 : B016 A00F 0140 2800 7800 1000 0180 2800
10 : 7800 1800 2800 7800 D016 B016 6800 7000
18 : 1840 2800 7800 0000 0000 0000 0000 0000
* D2 ↵ ←——— To check the contents of IM2

< DUMP IM2 >
00 : 001C 0018 6800 0170 915E D805 5200 0016
08 : 5400 0010 904C D60B 0028 003A C80A 800E
10 : D810 0039 C808 0018 914C 6A00 01D0 4A00
18 : 0005 D819 8009 800E 0018 5200 0016 5400
20 : 0010 D821 88BC D623 800E 5A00 00FF 1B2B
28 : 1B28 1798 CA2C 002C 0028 003A C821 0039
30 : C81F 0018 D832 915C 9040 D635 6800 1070
38 : 800E D839 905C 9820 DE3C C03D 0000 0000
40 : 0000 0000 0000 0000 0000 0000 0000 0000
48 : 0000 0000 0000 0000 0000 0000 0000 0000
50 : 0000 0000 0000 0000 0000 0000 0000 0000
58 : 0000 0000 0000 0000 0000 0000 0000 0000
60 : 0000 0000 0000 0000 0000 0000 0000 0000
68 : 0000 0000 0000 0000 0000 0000 0000 0000
70 : 0039 0000 800F E0A7 0000 E0A0 8400 DE77
78 : 8510 DE79 8610 C092 B020 E200 0000 0000
*
```

```
* CR GR0↵ ←———— The CR command is used to change the register contents
GR0   0020-0005↵
GR1   0014-0003↵
GR2   0000-/↵
* GO 24,2F,2 ↵←———— To execute the program
SET PARAMETER OR RETURN
? /D↵

D-PROCESSOR RESTARTS AFTER BREAK (Y OR OTHER)
? Y↵

SRP STARTS

< REGISTER DUMP >
IR    - FFFF  PC1   - 0023  GR0   - 0005  GR1   - 0002  GR2   - 0000
GR3   - 0001  GR4   - FF00  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0000
IOSR2 - 01D0  PCNTA - 0000  PCNTB - 0000  PC2F  - 00D3  BAR   - 090A
LCNT  - 140E  RCNT  - 0022  WCNTA - 0000  WCNTB - 0010

< REGISTER DUMP >
IR    - FFFF  PC1   - 0023  GR0   - 0015  GR1   - 0001  GR2   - 0000
GR3   - 0001  GR4   - FF10  GR5   - 00FF  GR6   - 0000  GR7   - 0000
GARG  - 0000  MC    - 0000  DRG0  - 0000  DRG1  - 0000  IOSR1 - 0000
IOSR2 - 01D0  PCNTA - 0000  PCNTB - 0000  PC2F  - 00D3  BAR   - 090A
LCNT  - 140E  RCNT  - 0022  WCNTA - 0000  WCNTB - 0010
IM-1 dump or IM-2 dump (1/2) ? 1↵
* D1 ↵ ←— To check the contents of IM1

< DUMP IM1 >
00 :  E000  2800  7800  0040  9001  8000  D009  C00A
08 :  B016  A00F  0140  2800  7800  1000  0180  2800
10 :  7800  1800  2800  7800  D016  B016  6800  7000
18 :  1840  2800  7800  0000  0000  0000  0000  0000
* D2↵ ←— To check the contents of IM2

< DUMP IM2 >
00 :  001C  0018  6800  0170  915E  D805  5200  0016
08 :  5400  0010  904C  D60B  0028  003A  C80A  800E
10 :  D810  0039  C808  0018  914C  6A00  01D0  4A00
18 :  0005  D819  8009  800E  0018  5200  0016  5400
20 :  0010  D821  88BC  D623  800E  5A00  00FF  1E2B
28 :  1B28  1798  CA2C  002C  0028  003A  C821  0039
30 :  C81F  0018  D832  915C  9040  D635  6800  1070
38 :  800E  D839  905C  9820  DE3C  C03D  0000  0000
40 :  0000  0000  0000  0000  0000  0000  0000  0000
48 :  0000  0000  0000  0000  0000  0000  0000  0000
50 :  0000  0000  0000  0000  0000  0000  0000  0000
58 :  0000  0000  0000  0000  0000  0000  0000  0000
60 :  0000  0000  0000  0000  0000  0000  0000  0000
68 :  0000  0000  0000  0000  0000  0000  0000  0000
70 :  0039  0000  800F  E0A7  0000  E0A0  8400  DE77
78 :  8510  DE79  8610  0000  B020  C024  0000  0000
*
```

Programs are developed in this manner.

CHAPTER 4 SIMPLE ASSEMBLER SYNTAX

4.1 Outline of Function

This is a 1-pass assembler for the μPD7764 which operates on the EVAKIT-7764 and converts the source.statement input from the console into the object codes. The object codes are stored in the effective addresses.

4.2 Characters

The assembler can recognize only hexadecimal numbers that indicate the reserved words of each field, immediate data, or address. The statement of the source program to be assembled by the assembler, however, can be described using the following characters.

(1) Alphabet

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
(Both the upper-case and lower-case characters can be used. The assembler does not distinguish between them.)

(2) Numerals

0 1 2 3 4 5 6 7 8 9

(3) Special characters

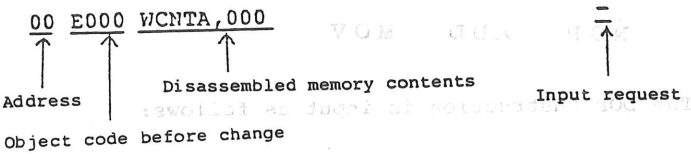
Δ (blank), : /↵(carriage return)
The functions of the special characters are listed in Table 4-1.

Table 4-1 Functions of Special Characters

Character	Name	Function
Δ	Blank	Field separator
,	Comma	Field separator
:	Colon	Same as CHANGE command
/	Slash	
↵	Carriage return	For details, refer to description of CHANGE command.

4.3 IM1 Assembler

The IM1 assembler is invoked by the C1X command. When the IM1 assembler is invoked by executing the C1X command, the following message is displayed on the console and the program enters the command input wait state.



After the hyphen, input the DOP or DLD instruction described in the following sections.

*For details on invoking the C1X command, refer to 3.3.2 (2), C1X command.

4.3.1 DOP instruction

The DOP instruction is input in the following format.

Input format:

DC field,S field,R field,ALU0 field,ALU1 field,ALU23 field

(When no instruction is specified, the NOP instruction is assumed in each field.)

The reserved words for each field are as follows:

*Reserved words for DC field

NOP DLJ DHJ RDF SDF MA
MAS ML0 ML0S ML2 ML2S RIOB
SIOB WIOB WAIT

*Reserved words for S field

NOP L0W2 A2B1 A2W1

*Reserved words for R field

NOP WB1 WB2 R RI RC1
RC2

*Reserved words for ALU0 field

NOP SUB SUBA SUBT

*Reserved words for ALU1 field

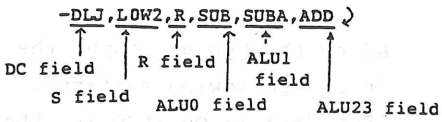
NOP SUB SUBA SUBT

*Reserved words for ALU23 field

NOP ADD MOV

The DOP instruction is input as follows:

```
00 E000 WCNTA,000
```



Delimit each field from the other by a comma (,).

*In case the DOP instruction is input in a wrong input format or characters other than the reserved words are input, a question mark (?) is displayed and the program reenters the command input wait state. If this happens, input the correct characters in the correct format.

4.3.2 DLD instruction

The DLD instruction is input in the following format.

Input format:

DEST field, nnn

nnn : Immediate data

*Reserved words for DEST field

RCNT	PC1	LCNTL	LCNTH
BARL	BARH	WCNTA	WCNTB

The DLD instruction is input as follows:

```
01 2800 SDF  ,NOP ,NOP ,NOP ,NOP ,NOP   -RCNT,005↵
                                          ↑    ↑
                                       DEST field  Immediate data
```

*In case the DLD instruction is input in a wrong input format or characters other than the reserved words are input, a question mark is displayed and the program reenters the command input wait state. If this happens, input the correct characters in the correct format.
If the immediate data exceeding the valid number of digits is specified, the latter 3 digits of the data are treated as valid. In contrast, if the data is less than the valid number of digits, the higher digit(s) is assumed to be 0.

4.4 IM2 Assembler

The IM2 assembler is invoked by the C2X command. When the IM2 assembler is invoked by executing the C2X command, the following message is displayed on the console and the program enters the command input wait state.

```
00 6800  0170  NOP    NON    ,IOSR1   ,0170  -
↑  ↑     ↑                   ↑                ↑
Address  Immediate data is displayed  Disassembled   Input request
         when the LD instruction is   memory contents
         specified.
Object code before change
```

After the hyphen, input the OP, LD, MOV, or JP instruction described in the following sections.
*For details on invoking the C2X command, refer to 3.3.2 (4), C2X command.

4.4.1 OP instruction

The OP instruction is input in the following format.

Input format:

ALU field△RS field,BS field

(If the ALU field is omitted, the NOP instruction is assumed. If the RS or BS field is omitted, the NON instruction is assumed.)

The reserved words for each field are as follows:

*Reserved words for ALU field

NOP	CLR	INC	DEC	CPL1
CPL2	SHRA	SHLA	SHR1	SHL1
SHR2	SHL2	SHR4	SHL4	SWAP
OR	AND	XOR	SUB	ADD
SBB	ADC	MIN	CMPR	SUBI
SUBP	ADDP	SBBP	ADCP	

*Reserved words for RS field

NON	GR0	GR1	GR2	GR3
GR4	GR5	GR6	GR7	

*Reserved words for BS field

NON	SR	DR	DRNF
PC1	IM1	IM2	GR0
GR1	GR2	GR3	GR4
GR5	GR6	GR7	GARG
MC	DRG0	DRG1	IOSR1
IOSR2	PCNTA	PCNTB	M
MCLR	MINC	MDEC	MBC
MBCCLR	MBCINC	MBCDEC	

The OP instruction is input as follows:

```
00 6800   0170   NOP ..  NON    ,IOSR1   ,0170   -ADD GR0,GR2
                                                   ↑     ↑    ↑
                                             ALU field   |   BS field
                                                         |
                                                     RS field
```

*In case the OP instruction is input in a wrong input format or characters other than the reserved words are input, a question mark is displayed and the program reenters the command input wait state. If this happens, input the correct characters in the correct format.

4.4.2 LD instruction

If immediate data is written after the BS field of the OP instruction, the OP instruction serves as the LD instruction. The LD instruction is input in the following format.

Input format:

ALU field△RS field,BS field,nnnn

nnnn : Immediate data

If the ALU field and RS field are omitted, the statement is treated as the MVI instruction. The MVI instruction is input in the following format.

Input format:

MVI△BS field,nnnn

nnnn : Immediate data

(When the MVI instruction is specified, the ALU field is processed as the NOP instruction and RS field is processed as the NON instruction.) The reserved words for each field of the LD and MVI instructions are the same as those of the OP instruction. For details on the reserved words, refer to the description of the OP instruction.

The LD instruction is input as follows:

```
0C 0028        INC    GR0    ,NON          →CLR△GR1,GR2,0021 ↵
                                             ↑    ↑   ↑    ↑
                                        ALU field │   BS field
                                             RS field
                                                  Immediate data
```

The MVI instruction is input as follows:

```
11 0039        DEC    GR1    ,NON          -MVI△GR0,0001↵
                                                ↑      ↑
                                              BS field  Immediate data
```

*In case the LD or MVI instruction is input in a wrong input format or characters other than the reserved words are input, a question mark is displayed and the program reenters the command input wait state. If this happens, input the correct characters in the correct input format. If the immediate data exceeding the valid number of digits is specified, the latter 3 digits of the data are treated as valid. In contrast, if the data is less than the valid number of digits, the higher digit(s) is assumed to be 0.

4.4.3 MOV instruction

The MOV instruction is input in the following format.

Input format:

MOV△DST field,SRC field,GM field

(If the DST or SRC field is omitted, the NON instruction is assumed. If the GM field is omitted, the NOP instruction is assumed.

The reserved words for each field are as follows:

*Reserved words for SRC and GM fields

NON	SR	DR	DRNF
PC1	IM1	IM2	GR0
GR1	GR2	GR3	GR4
GR5	GR6	GR7	GARG
MC	DRG0	DRG1	IOSR1
IOSR2	PCNTA	PCNTB	M
MCLR	MINC	MDEC	MBC
MBCCLR	MBCINC	MBCDEC	

*Reserved words for GM field

NOP	LOW	HIGH	WRD	RSL
SSL	BST2	BST3	RIOB	SIOB
RINTR	SINTR	RDF	STTD	STPD

The MOV instruction is input as follows:

```
13 0018        CLR    GR0    ,NON          -MOV IOSR1,GR0,RDF ↵
                                                 ↑    ↑    ↑
                                                DST  SRC  GM field
```

*In case the MOV instruction is input in a wrong input format or characters other than the reserved words are input, a question mark is displayed and the program reenters the command input wait state. If this happens, input the correct characters in the correct input format.

4.4.4 JP instruction

The following three input formats are available for the JP instruction.

(1) JUMP instruction

Input format:

JMP△CND field,nn

nn: Jump destination address (in hexadecimal format)

(If the CND field is omitted, the unconditional JUMP instruction is assumed.)

(2) CALL instruction

Input format:

CALL△CND field,nn

nn : Jump destination address (in hexadecimal format)

(If the CND field is omitted, the unconditional CALL instruction is assumed.)

(3) RETURN instruction

Input format:

RET (Returns flag)

RETR (Does not return flag)

*Reserved words for CNT field

NC	C	NZ	Z	NOV
OV	P	M	NIOB	IOB
NDF	DF	NRQM	RQM	

The JUMP instruction is input as follows:

```
04 915E        MOV    IOSR2 ,GR0     ,STTD  -JMP NZ,01↵
                                                 |  |
                                                 |  Destination address
                                                 CND field
```

Input example when the CNT field is omitted

```
1A 8009        MOV    NON   ,NON     ,SIOB  -JMP ,10↵
                                                 |
                                                 Destination address
                                    Be sure to input a comma here.
```

The CALL instruction is input as follows:

```
2A CA2C        JMP    Z     ,2C              -CALL NIOB,5F↵
                                                  |    |
                                                  |    Destination address
                                                  CND field
```

Input example when the CNT field is omitted

```
2B 002C        INC    GR4    ,NON          -CALL ,60↵
                                                ↑
                                          Destination
                                          address
```

Be sure to input a comma here.

The RETURN instruction is input as follows:

.To return flag

```
2C 0039        DEC    GR1    ,NON          -RET↵
```

.To not return flag

```
23 CA25        JMP    Z      ,25           -RETR↵
```

*In case the JUMP instruction is input in a wrong input format or characters other than the reserved words are input, a question mark is displayed and the program reenters the command input wait state. If this happens, input the correct characters in the correct input format. If the immediate data exceeding the valid number of digits is specified, the latter 3 digits of the data are treated as valid. In contrast, if the data is less than the valid number of digits, the higher digit(s) is assumed to be 0.

APPENDIX 1 CPU BOARD JUMPER SETTING

The jumpers on the EVAKIT-7764 CPU board are set as shown in the following table.

CPU Board Jumper Setting

Jumper	Function	Setting	Remarks
JP0	CPU clock	2-3	3.93216MHz
JP1,JP2	BCLK, CCLK	Short	Supplied by CPU board
JP3	CPU clock	1-2	3.93216MHz
JP4	Bus time out	Open	Not forcibly enabled
JP5	ROM/RAM selection	2-5 3-4	IC34,IC46 µPD2764 IC35,IC47 µPD2764
JP6		Open	
JP7		1-2	
JP8		Open	
JP9	RAM power source	2-3	Supplied by CPU board
JP10	RAM type selection	Short	µPD446
JP11		3-11	
JP12	RAM address	2-3	A000H to AFFFH
JP13	TIMER clock	1-8	1.2288MHz µPD8253 CLK2
JP14	Interrupt	1-11	NMI-GND
JP15		1-2	RS-232C (TxD-J3 PIN22)
JP16		1-2	RS-232C (RxD-J3 PIN24)
JP17		1-2	
JP18		1-4	RS-232C (CTS)
JP19	SIO interface selection	1-4	RTS-J3 PIN18
		2-5	CTS-J3 PIN20
JP20		2-3-4	TxC, RxC µPDF8253 2 outputs
JP21		1-3	DSR-J3 PIN13
		2-4	DTR-J3 PIN16
JP22		Open	No power supplied to J3
JP23		Short	J3 PIN26-GND

Jumper	Function	Setting	Remarks
JP24 JP29	PIO	Open	PIO unused
JP30	CBRQ	2-3	No output to multibus
JP31	BANK changeover wait	1-2 (on PWB)	

NOTE: These jumpers are correctly factory-set for shipment. Do not change their settings.

APPENDIX 2 FILE FORMAT

The SAVE and LOAD commands of the EVAKIT-7764 use the following file format. This file format is the HEX format of INTEL.

Format:

```
': CCAAAATTH1H2 ........HnSSCrLf'
```

':'	Record mark (Indicates the beginning of a record.)
CC	Number of codes (Written as a 2-digit hexadecimal number and indicates the number of object codes that can be written per record. The maximum number of codes is 10H. 00H indicates the end of the record.)
AAAA	Location address (Written as a 4-digit hexadecimal number and indicates the first address of a record.)
TT	Record type (Written as 2-digit hexadecimal number. 01 indicates the end record and 00 is assigned to the other records.)
H1 to Hn	Code (Written as 2-digit hexadecimal code and indicates one word by one byte.)
SS	Checksum (The number of codes, location address, record type, and code are successively subtracted from initial value 00. The value obtained as a result of the subtraction is indicated here as a 2-digit hexadecimal number.)
Cr	Carriage return (0DH)
Lf	Line feed (0AH)

NOTE: No meaningless data must be placed before ':'.

Although the record format processed by this monitor is in the HEX format of INTEL as shown above, the three types of files described below regard one physical record as 128 bytes. In one physical record, HEX-format object frames are placed in series and the frames exceeding the record are placed in the next record from the beginning of the next record.

After the last object frame, the following frame is required as the end frame.

```
':00000001FFCrLf'
```

If a vacant frame follows the end frame, insert dummy data in the vacant frame.

(1) Instruction RAM file

This file is used by the L, L1, L2, and S commands and can use the object program files (HEX format) output by the CP/M version μPD7764 assembler.

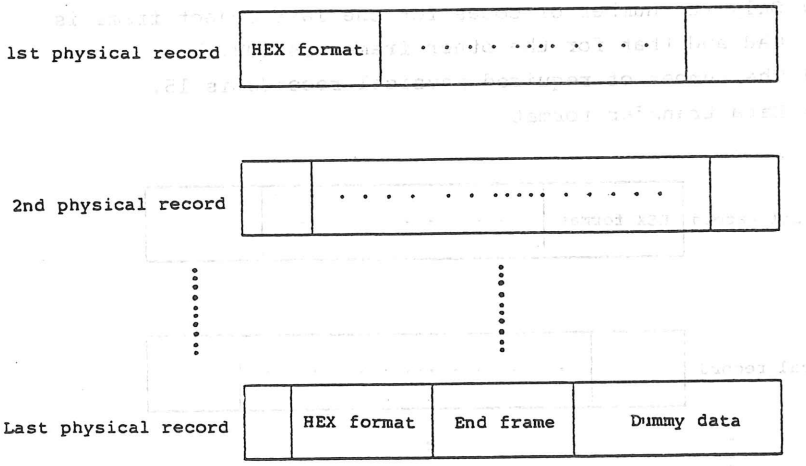
The details of the file are as follows:

(a) The number of data for binary image to be transferred is 320 bytes (from address 00 of IM1 to address 7F of IM2).

(b) Only the number of codes for the object frame of the LD instruction is 04H and that for the other frames is 02H.

(c) The maximum number of records required is 22.

(d) Data transfer format



(2) Data RAM file and internal register file

These files are used by the LW and SW commands.
The details of this file are as follows:

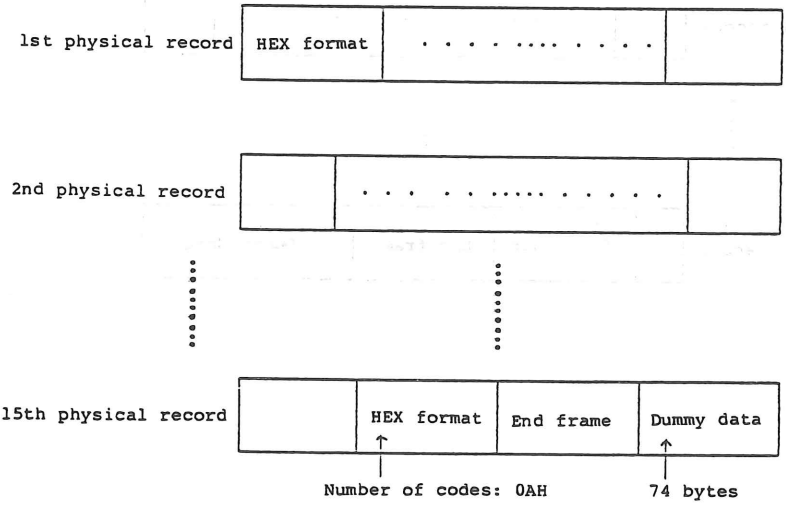
(a) The number of data for the binary image to be transferred is the following 650 bytes.

System Constant (00H,00H,83H,02H,06H,00H) : 6 bytes
RAM-A (Bank0H,address0H to bank15H,addressFH):352 bytes
RAM-B (Bank0H,address0H to bank3H,addressFH) : 64 bytes
RAM-G (Bank0H,address0H to bank3H,address15H):176 bytes
Internal register (Dummy,dummy,IR,PC1,GR0 to GR7,GARG, MC,DRG0,DRG1,IOSR1,IOSR2,PCNTA, PCNTB,PC2F,BAR,LCNT,RCNT,WCNTA, WCNTB) : 52 bytes
Total:650 bytes

(b) Only the number of codes for the last object frame is 0AH and that for the other frames is 10H.

(c) The number of required physical records is 15.

(d) Data transfer format



(3) Pattern memory file

This file is used by the LP and SP commands.

The details of this file are as follows:

(a) The number of data for the binary image to be transferred is the contents of the pattern memory shown below ((end address + start address) + 1) and 6 bytes (Sl to Bh).

SlShElEhBlBhDlD2 ..... De

Sl : Lower 1 byte of the pattern memory start address

Sh : Higher 1 byte of the pattern memory start address

El : Lower 1 byte of the pattern memory end address

Eh : Higher 1 byte of the pattern memory end address

Bl : Lower 1 byte of the number of records

Bh : Higher 1 byte of the number of records

Dl to D6: Pattern memory contents

The number of data can be calculated by this expression.

Number of data = (End address - Start address) + 7

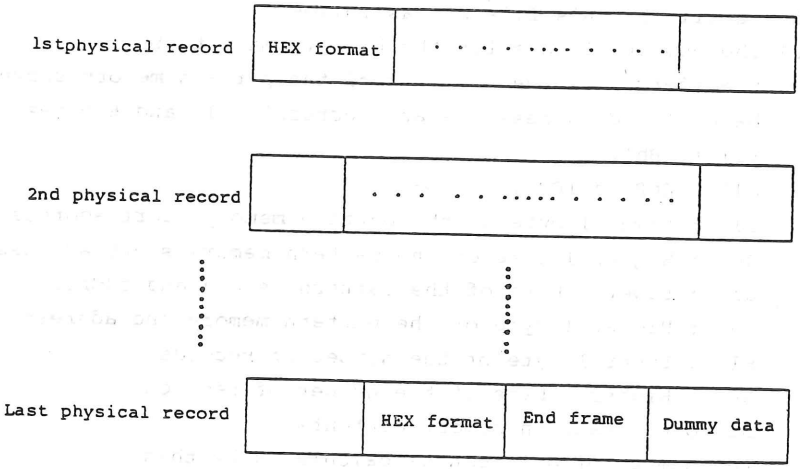
(b) The number of codes for the last HEX object frame is the number of data and that for the other frames is 10H.

(c) The required number of physical can be calculated by this expression.

Number of physical records

= {(Number of data ÷ 16) x 45} ÷ 128

(d) Data transfer format



*Of 16 data in the HEX-format object frame of the first physical record, data from S1 to Bh in (a) are placed in the first 6 data.

APPENDIX 3 RESERVED WORDS FOR OPERAND DESCRIPTION

(List of Mnemonics)

Mnemonic	Specified register
NON	NO register
SR	SR register
DR	DR register
DRNF	DR register (RQM flag is not set.)
IR	IR register
PC1	PC−1
IM1	Inst. Mem.−1
IM2	Inst. Mem.−2
GR0	GR0 register
GR1	GR1 register
GR2	GR2 register
GR3	GR3 register
GR4	GR4 register
GR5	GR5 register
GR6	GR6 register
GR7	GR7 register
GARG	GARG register
MC	MC register
DRG0	DRG0 register
DRG1	DRG1 register
IOSR1	IOSR−1
IOSR2	IOSR−2
PCNTA	PCNT−A
PCNTB	PCNT−B
M	No Operation
MCLR	Clear Index.
MINC	Increment Index
MDEC	Decrement Index
MBC	$\overline{L}$ to L
MBCCLR	$\overline{L}$ to L . Clear Index
MBCINC	$\overline{L}$ to L . Increment Index
MBCDEC	$\overline{L}$ to L . Decrement Index

Mnemonic	Function
NOP	No Operation
CLR	Clear
INC	Increment
DEC	Decrement
CPL1	1's Complement
CPL2	2's Complement
SHRA	Arithmetical R-Shift
SHLA	1 Bit L-Shift Without Carry
SHR1	1 Bit R-Shift
SHL1	1 Bit L-Shift
SHR2	2 Bit R-Shift
SHL2	2 Bit L-Shift
SHR4	4 Bit R-Shift
SHL4	4 Bit L-Shift
SWAP	Swap
OR	OR
AND	AND
XOR	Exclusive OR
SUB	Subtract
ADD	Add
SBB	Subtract With Borrow
ADC	Add With Carry
MIN	Min
CMPR	Compare
SUBI	SUBI
SUBP	Protect SUB
ADDP	Protect ADD
SBBP	Protect SBB
ADCP	Protect ADC

RS field

Mnemonic	Specified register
NON	NO register
GR0	GR0 register
GR1	GR1 register
GR2	GR2 register
GR3	GR3 register
GR4	GR4 register
GR5	GR5 register
GR6	GR6 register
GR7	GR7 register

④ CM field

Mnemonic	Function
NOP	No Operation
LOW	RAM-G Lower Byte Access
HIGH	RAM-G Higher Byte Access
WRD	RAM-G Word Access
RSL	SL Reset
SSL	SL Set
BST2	Bank Set 2
BST3	Bank Set 3
RIOB	Reset IOB
SIOB	Set IOB
RINTR	INTR Reset
SINTR	INTR Set
RDF	DF Reset
STTD	Start D Processor
STPD	Stop D Processor

CD field

Mnemonic	Condition
NC	C = 0
C	C = 1
NZ	Z = 0
Z	Z = 1
NOV	OV = 0
OV	OV = 1
P	S = 0
M	S = 1
NIOB	IOB = 0
IOB	IOB = 1
NDF	DF = 0
DF	DF = 1
NRQM	RQM = 0
RQM	RQM = 0

⑥ DC field

Mnemonic	Function
NOP	No Operation
DLJ	Decrement $LCNT_L$ and Test
DHJ	Decrement $LCNT_H$ and Test
RDF	Reset DF
SDF	Set DF
MA	A1 to DRG0
MAS	A1 to DRG0 , Set DF
ML0	L0 to DRG0
ML0S	L0 to DRG0 , Set DF
ML2	L2 to DRG0
ML2S	L2 to DRG0 , Set DF
RIOB	Reset IOB
SIOB	Set IOB
WIOB	Wait until IOB=0
WAIT	Wait

⑦ S field

Mnemonic	Function
NOP	No Operation
L0W2	L0 to P1 . W2 to Q1
A2B1	A2 to P1 . B1 to Q1
A2W1	A2 to P1 . W1 to Q1

⑧ R field

Mnemonic	Function
NOP	No Operation
WB1	$(WCNT-B)_H \leftarrow \div 1$
WB2	$(WCNT-B)_H \leftarrow \div 2$
R	READ
RI	READ. Increment $(RCNT)_L$
RC1	READ. Clear $(RCNT)_L$. $(RCNT)_H \leftarrow \div 1$
RC2	READ. Clear $(RCNT)_L$. $(RCNT)_H \leftarrow \div 2$

⑨ ALU0 and ALU1 fields

Mnemonic	Function
NOP	No Operation
SUB	Subtract
SUBA	SUB & ABS
SUBT	SUB & TEST

⑩ ALU23 field

Mnemonic	Function
NOP	No Operation
ADD	Add
MOV	MOVE

(11) DEST field

Mnemonic	Specified register
RCNT	RCNT
PC1	PC-1
LCNTL	LCNT (Lower 8 bits)
LCNTH	LCNT (Higher 8 bits)
BARL	BAR (Lower 8 bits)
BARH	BAR (Higher 8 bits)
WCNTA	WCNT-A
WCNTB	WCNT-B

APPENDIX 4 LIST OF ERROR MESSAGES

Message: COMMAND ERROR	
Command	All commands
Cause	Command input error
Monitor's operation	Displays "*" and waits for command input.
User's action	Input the correct command.
Message: OPERAND ERROR	
Command	All commands
Cause	An error in the input format of bank or address
Monitor's operation	Displays "*" and waits for command input.
User's action	Input the correct bank or address after the command.
Message: INHIBIT LOAD COMMAND AT 7F ADDRESS !	
Command	C2X
Cause	An attempt is made to change address 7F (end address).
Monitor's operation	Requests for reinput.
User's action	Use an instruction other than LD.
Message: PARAMETER ERROR	
Command	GO
Cause	Input parameter is wrong.
Monitor's operation	Requests for reinput.
User's action	Input the correct parameter.
Message: DATA ERROR	
Command	GO
Cause	Data input error
Monitor's operation	Requests for reinput.
User's action	Input the correct data.

Message: BREAK POINT IS JMP COMMAND	
Command	GO (when the break point is set)
Cause	The break point is set at the address where the JMP instruction is stored.
Monitor's operation	Displays "*" and waits for command input.
User's action	Set the break point at an address where the JMP instruction is not stored.
Message: BREAK POINT IS RET COMMAND	
Command	GO (when the break point is set)
Cause	The break point is set at the address where the RET instruction is stored.
Monitor's operation	Displays "*" and waits for command input.
User's action	Set the break point at an address where the break point is not set.
Message: MODE ERROR	
Command	GO (when the break point is set)
Cause	The program execution passes the break point with the ROM flag set to 11.
Monitor's operation	Displays "*" and waits for command input.
User's action	Change the set address where the break point is set.

Message: HARD ERROR	
Command	All commands
Cause	1. The EVAKIT-7764 is not correctly connected to the user system. 2. The μPD7764 is not correctly inserted into the socket or is malfunctioning.
Monitor's operation	Displays "*" and waits for command input.
User's action	1. Check the connection. 2. Check the μPD7764 for proper insertion into the socket or for malfunction

NEC OFFICES

NEC Electronics (Europe) GmbH, Oberrather Str. 4, 4000 Düsseldorf 30, W.-Germany, Tel. (0211) 650301, Telex 858996-0

NEC Electronics (Germany) GmbH, Kanzlerstr. 6, 4000 Düsseldorf 30, Tel. (0211) 650302, Telex 858996-0
- Hindenburgstr. 28/29, 3000 Hannover 1, Tel. (0511) 881013-16, Telex 9230109
- Arabellastr. 17, 8000 München 81, Tel. (089) 416020, Telex 522971
- Heilbronner Str. 314, 7000 Stuttgart 30, Tel. (0711) 890910, Telex 7252220
- NEC Electronics (BNL) – 33 Alard du Hamelstraat, 5622 CC Eindhoven, Tel. (040) 445845, Telex 51923
- NEC Electronics (Scandinavia) – Box 4039, S-18304 Täby, Tel. (08) 756 7245, Telex 13839

NEC Electronics (France) S.A., Tour Chenonceaux, 204, Rond Point du Pont de Sèvres, F-92516 Boulogne Billancourt, Tel. (01) 609 9004, Telex 203544

NEC Electronics Italiana S.R.L., Via Cardano 3, I-20124 Milano, Tel. (02) 67 09 108, Telex 315355

NEC Electronics (UK) Ltd., Block 3 Carfin Industrial, Motherwell M L1 4UL, Schottland, Tel. (0698) 732221, Telex 777565